Difference between file system and DBMS

Diff" factor	file system (ext 3, ext 4, ntfs, btfs)	DBMS (sqlite, mysql, postgres, mongo, oracle)
defn	is abstraction which helps users to perform manipulation on different types of files.	is software package which allows user to create, update and delete from DB.
data redundancy	in file systems redundancy is more as compared to DBMS	less redundancy is provided with the help of DBMS as compared to file systems.
data consistency	are less consistent as compared to DBMS.	are more consistent because of less redundancy.
easy to access	are more difficult to access as compared to DBMS.	accessibility is faster as query tuning (SQL) have been provided by DBMS.
sharing of data	data is not centrally located as it is scattered at multiple places, sharing info. is difficult.	all the data sets are centrally located and therefore information can be shared easily.
for DB e.g.:- (password protection, face, voice detection, fingerprint matching) Security	no encryption methods were introduced earlier in early days, therefore security was a major concern.	with development, various encryption methods are introduced.
data integrity	There is no mechanism for checking values before entering in the datasets.	Checks constraints are provided so that no wrong data can be entered.
Atomicity (interaction)	atomicity is absent in filesystems.	supports atomicity
Concurrency	not supported.	concurrency is provided.
Data isolation	Database	

→ next page

data redundancy eg:-

150 – comps
150 – I.T

S_rollno name contact no email id DoB branch HOD name HOD contact HOD email

300 × 9 = 2700
rows col.

Table 1

S_rollno name contact no email id DoB branch id

300 × 36 = 10800

Table 2

Branch_id branch HOD name contact email

2 × 5 = 10

1810

data consistency eg:-

If we consider above student, we want to make change to all 150 rows from ABC to XYZ. If we forget to make change in two places, one will have 2 value for HOD name therefore it will result in inconsistent state.

Data isolation → Due to scattered datasets, data can be stored using different file formats and hence data isolation is the problem. (for file system)

for DB { Everything is present at one server and therefore datasets are stored in same file format.

Data integrity → Whatever data we enter must be accurate and should follow certain rules. eg:- age of student must be > 20 for admission to an institute.

very interesting Atomicity → says all the transactions must be completed successfully or none of them can be completed.

eg:- In banking transaction if there is a real time system error, with the property of atomicity prevents DB from having data in an inconsistent state.

Concurrency → several transactions are trying to access same data item at the same time.

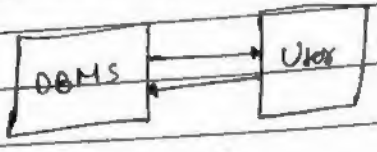
eg:- DB can use semaphores as a locking mechanism. The problem is similar to readers-writers problem from O.S.

diff. between semaphores and mutex
- mutex is a locking mechanism
- semaphore is a signalling mechanism.

Architecture of DBMS

It describes how DB components are organized and integrated. Types of DB architecture :-

1 tier arch.

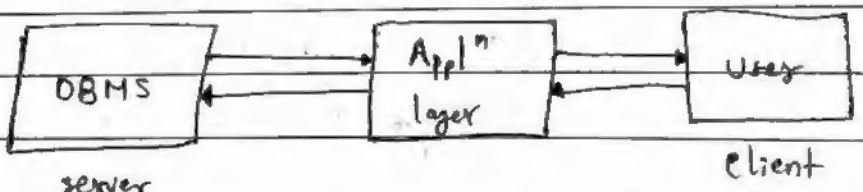


User is directly connected to DBMS and therefore all the changes can be made directly.

Adv. → simple design

Disadv. → security is a problem, any user can come and make changes.

2 tier arch.



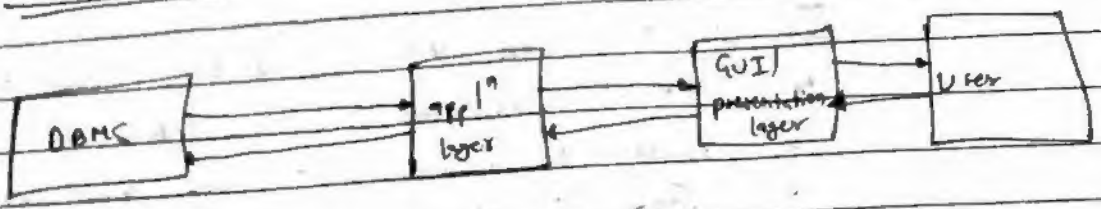
User has to communicate to the appln layer with DBMS.

→ provides set of programs for comm^n. eg:- JDBC - Java Database Connectivity, ODBC - Open Database Connectivity

↳ ASP.net

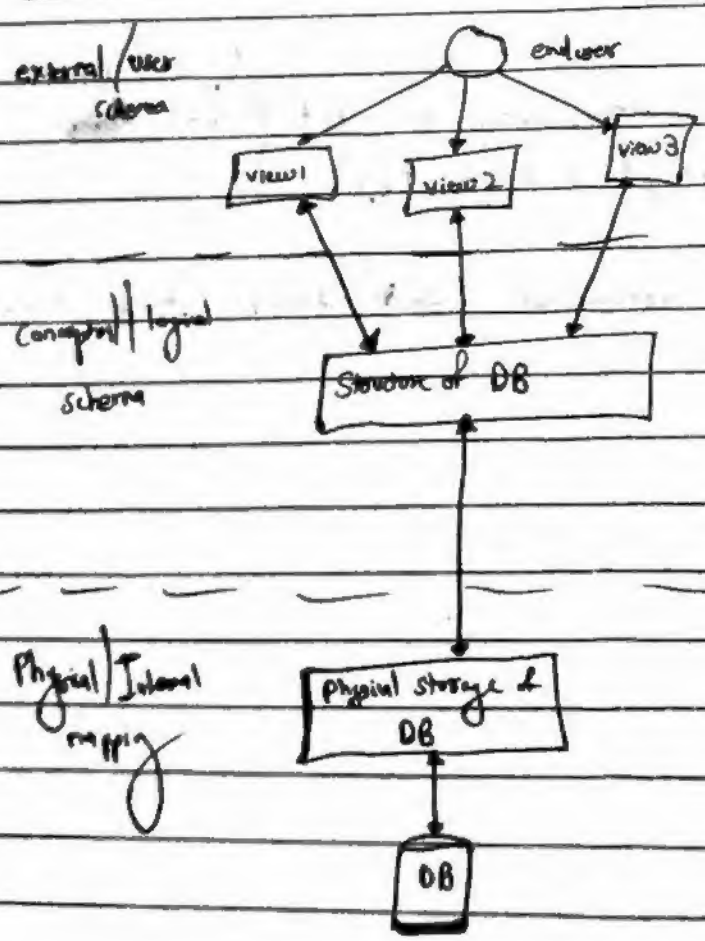
Adv. → security is more as compared to 1 tier arch.

3 tier arch.



User has to communicate with the GUI with the DB. User as with various apps as a presentation layer and request is given to the appl^n layer which will contact DBMS to provide an arch.

Three level architecture or Three views of database



external/conceptual mapping

request is sent by external schema.

response is sent by conceptual schema.

conceptual/internal mapping

External schema / user's view

It describes user view for the DB. Different users will have different perspectives for the DB.

eg:- According to university, they keep views such as Library and Student (Class advisor) views, with different attributes for both these views.

conceptual / logical schema

It defines structure of DB (no. of entities, attributes, datatypes and data structures).

Physical schema / internal schema

It defines physical representation of DB (how DB is being stored in a computer, size of DB, etc).

Schema → An arrangement which describes how datavalues are stored.

Data independence

It is an ability to modify one level without affecting next higher level.
It gets divided into 2 parts:—

- logical data independence
- physical data independence

logical data independence → it provides an ability to change structure of DB without affecting user's views.
eg:- we can add 'n' no. of records to table, only when info. is related to that particular table.

physical data independence → it provides ability to change structure of DB without affecting logical or user's views of DB.
eg:- moving a file from one location to another (physical storage location) without changing the structure of DB.

Users involved in DB

DBA (Database Administrator): A person (group of persons) centrally located having wide views required to run DB smoothly.
Such a person is appointed as DB administrator.
Roles performed by DBA:
① Developing conceptual schema → defines structure of DB
② Deciding which DB to use
↳ to create system in-house or use ready made tool
③ Defining DB and loading content
↳ (actual implementation) takes place.

④ Assisting and approving appln accesses
↳ helps programmers to write programs
⑤ Deciding Data structure.
↳ Continuous improvement for changing DB.
⑥ Backup and recovery
↳ recover from failures using backups.
⑦ Integrity
↳ checking data correctness in the DB.

② DB designers
③ Security officers
④ End users

DB designers → helps DBA in designing conceptual schema.
their job is to meet users and find out their requirements.
they will check whether all the user requirements are fulfilled or not.

Security officers → prevents unauthorized access to the DB.

End user → occasionally accessing database → Casual user
eg:- librarian of a college

Naive/parametric users → frequently accessing DB
eg:- clerks working at ATM transactions

Sophisticated users → learning from provided developed DBMS tools
and modifying them
eg:- business analysts, engineers

Standalone users → use readymade DBMS tools.
eg:- librarian of a college

Characteristics of DB

1. self describing nature of DB and metadata

metadata → data about data or data dictionary.
Keyword :- user manual for the DB.
eg:- data is copied from website source hence self describing nature and metadata is required.

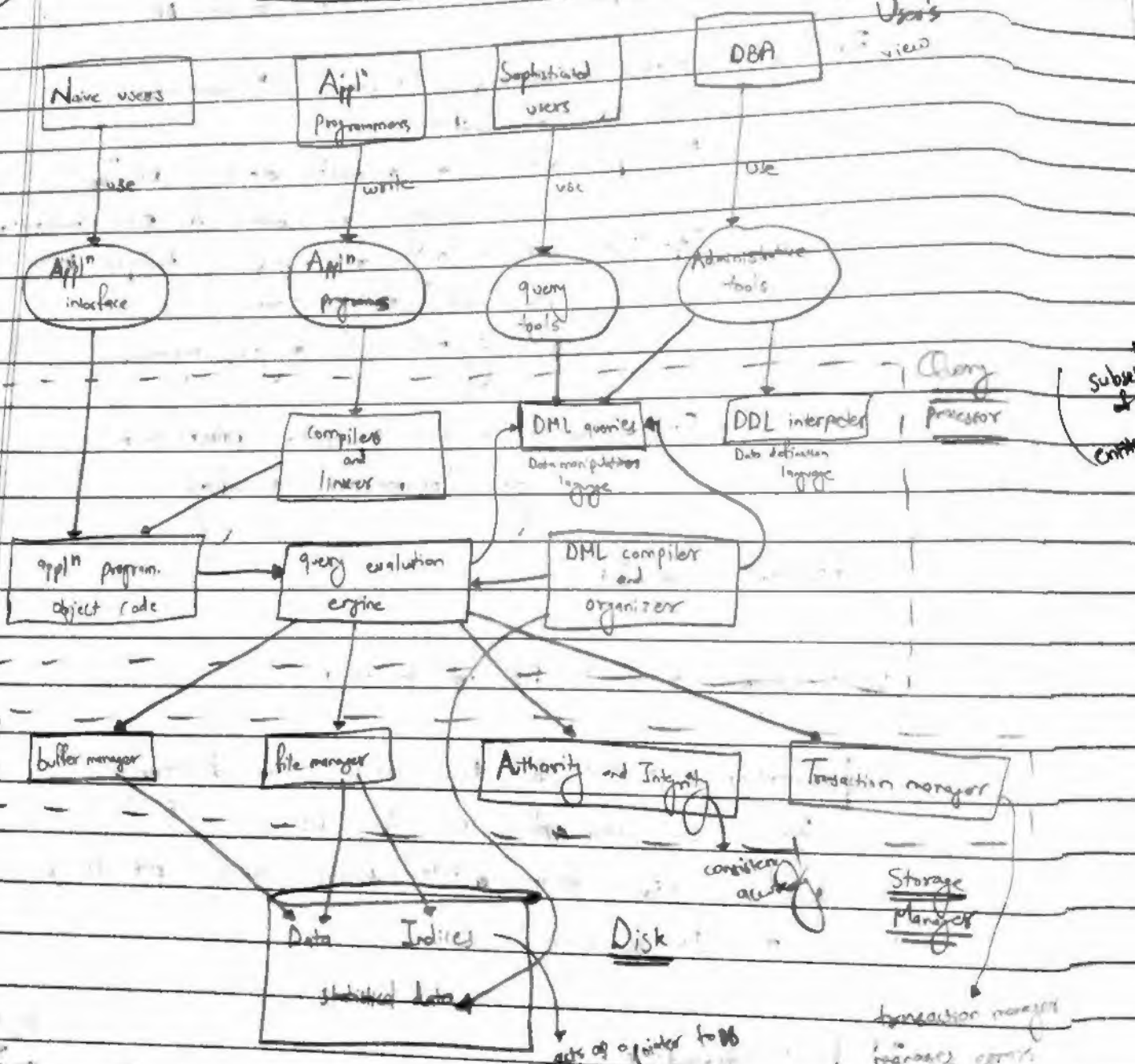
2. Data independence.

3. Multiple views of DB and security → different encryption methods
keyword :- diff. users going to use DB differently.

④ Data Persistance

Data stored in DB remains as it is even if power failure occurs or system gets crashed.

DBMS System Architecture



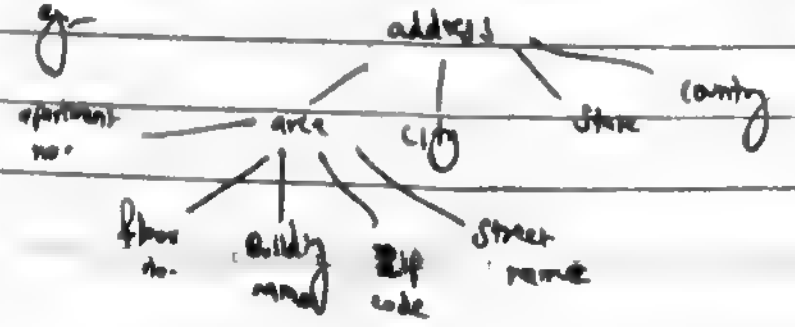
Entity Relationship Data Model

- ER Enterprise → nothing but an organization. eg:- hospital, bank, college, railway reservation system.
- Entity → any real world object is termed as entity. (person, place, event, thing).
- Attributes → Special characteristics which describe entity in detail.
  eg:- student roll_no, name, age, address.
- Entity type → collection of entities having common attribute.
  eg:- student information.
- Entity set → collection of entities (one or more than one) having same entity type, termed as entity set.
  (Subset of entities)
- Value → any information or data stored in a table is termed as value.
- Domain/Value set → set of values for particular attribute termed as domain or value set.

Types of attributes

- Single vs Composite
  - Single → this attribute cannot be divided further. eg:- color of car, age, roll no. of student.

Composite → attribute which can be further be divided in subparts.



- Single vs. multivalued
  - the attribute which will have only a single value.
    eg:- branch of student, roll no., age, height.
  - multivalued — attribute which can have multiple values.
    eg:- name (first name, middle name, surname)
    email id,
    contact no.

- stored vs. derived
  - 'age' attribute can be calculated if we know D.O.B of the student and current date.
    And therefore age is termed as derived attribute, which gets derived from one stored attribute i.e. D.O.B.

- Null value attribute
  Two cases:-
  1. Value does not exist
  2. Value exists, but not known.

Primary key cannot be empty or null

Keys in DBMS

Key is an attribute (or set of attributes) created in such a way that all records of a table are identified uniquely.

- Super key → super set of all the keys which will give us uniqueness.
- Candidate key → minimal super key (minimal → minimum set are used to make the key, by removing redundancy).
- primary key → any candidate key can be chosen by the developer for unique identification acts as a primary key.
- secondary/alternate key → excluding primary key, all other candidate keys are alternate keys.
- foreign key → primary key of one table acts as a normal col" in second table.

employee data

empid	fname	lname	sal	DOB	DOJ	passport no	Dept_Id
1	Priyanka	bbb	75k	13/2/90	1/2/12	-123-	10
2	Ankita	ccc	80k	2/12/91	1/2/12	-456-	20
3	Priyanka	bbb	85k	13/2/90	4/2/13	-789-	30
4	Tejaswi	aaa	80k	15/6/85	5/2/15	-023-	10

(foreign key → dept_Id; primary key → emp_id)

Dept data

Dept_Id	Dept name
10	Sales
20	Marketing
20	accounts

(✓ → assigned candidate keys; ⊗ → removed choice from candidate keys)

Super keys:- ① emp_Id ② emp_Id, fname ③ emp_Id, lname ④ fname, lname, sal (from fname, lname, sal) ⑤ passport no ⑥ passport no, dept_Id . . . . . .

Q.	Book_id	name	author
	B_1	XYZ	A_1
	B_2	ABC	A_{12}
	B_3	XYZ	A_2
	B_4	PQR	A_3
	B_5	LSP	A_4
	B_6	ABC	A_{13}

Super keys → book_id
book_id, name
book_id, author
name, author
book_id, name, author

Candidate keys → book_id
name, author

Primary key → book_id Secondary key → name, author

DATE / / /

Types of entity sets

① Strong entity set
② Weak entity set

① Strong entity set → if an entity set has a primary key attribute, it is termed as strong entity set.
eg:- Student (where Roll no. can be primary key)

② Weak entity set → if an entity set does not have any primary key, it is termed as weak entity set.
eg:- table (shape, size, colour, height) (No primary key)

Relationships

It describes an association among several entities.

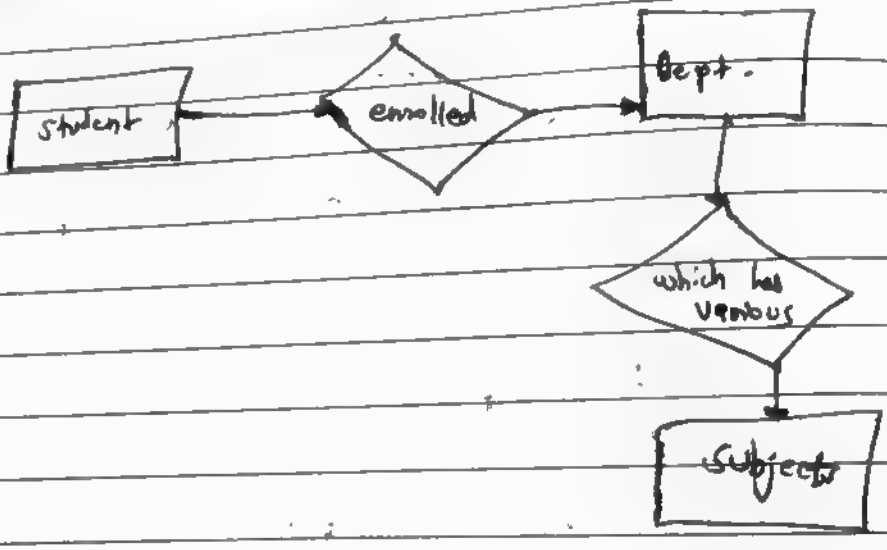
Degree of relationship

→ the no of entities involved in a comm^n describes the degree of relationships.

Unary relationship set → In this only one entity set participates in the comm^n.

Binary relationship set → In which two entity sets are involved.

Ternary relationship set → Student enrolled a department, which has various subjects.



Nary rel" set → More than 3 entity sets are involved.
eg: Prof. teaches in school, college and class.

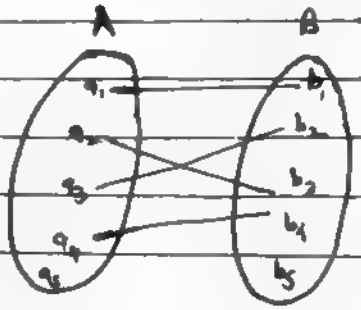
DATE / /

Can we have zero comm^n ?
→ Yes

Mapping constraints / Cardinality

① one to one (1:1)

Any entity in A is associated with atmost (1 or 0) 1 entity in B.
Any entity in B is associated with atmost 1 entity in A.

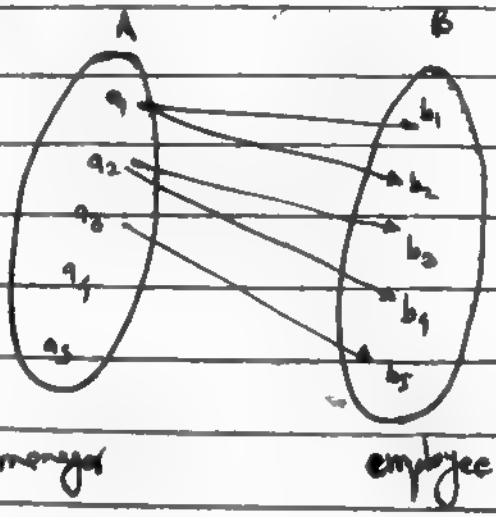


Country — Prime minister

② one to many (1:N)

A B

Any entity in A → associated with → any no. of (0 or more) entities in B
Any entity in B → atmost 1 entity in A

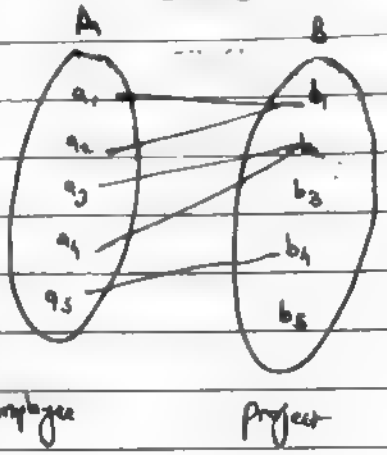


manager — employee

③ many to one (M : 1)

Any entity in A → atmost one entity in B

Any entity in B → any entity in A

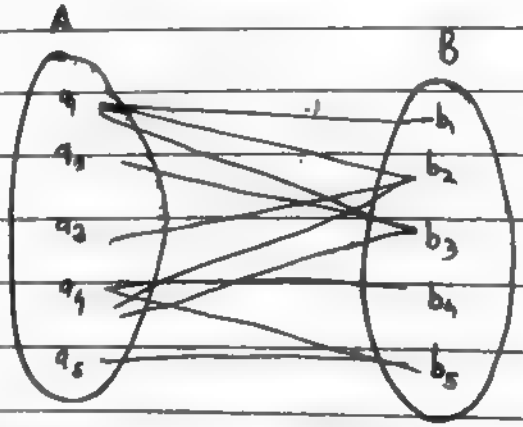


employee Project

④ many to many (M : N)

Any entity in A → any entity in B

and vice versa



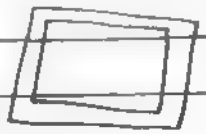
Student Teachers

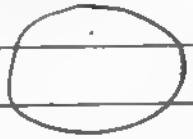
ER model

It is entity - rel" diagram which describes logical representation of DB.

Symbols used in ER diagram are as follows :-

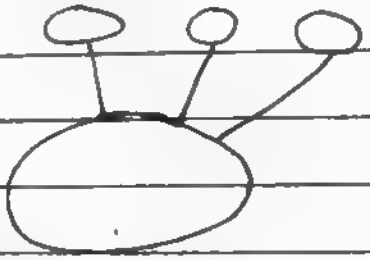
1.  → Strong entity set

2.  → weak entity set

3.  → attribute

4.  → Primary key attribute

5.  → multivalued

6.  → Composite attr.

7.  → derived attr.

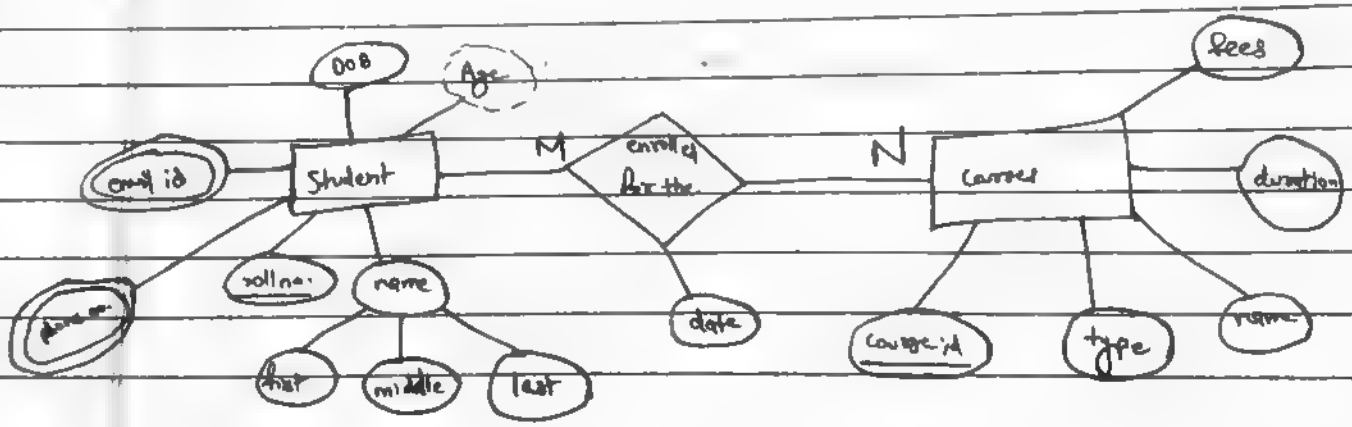
8. 

Total participation

partial participation

generalization or specialization

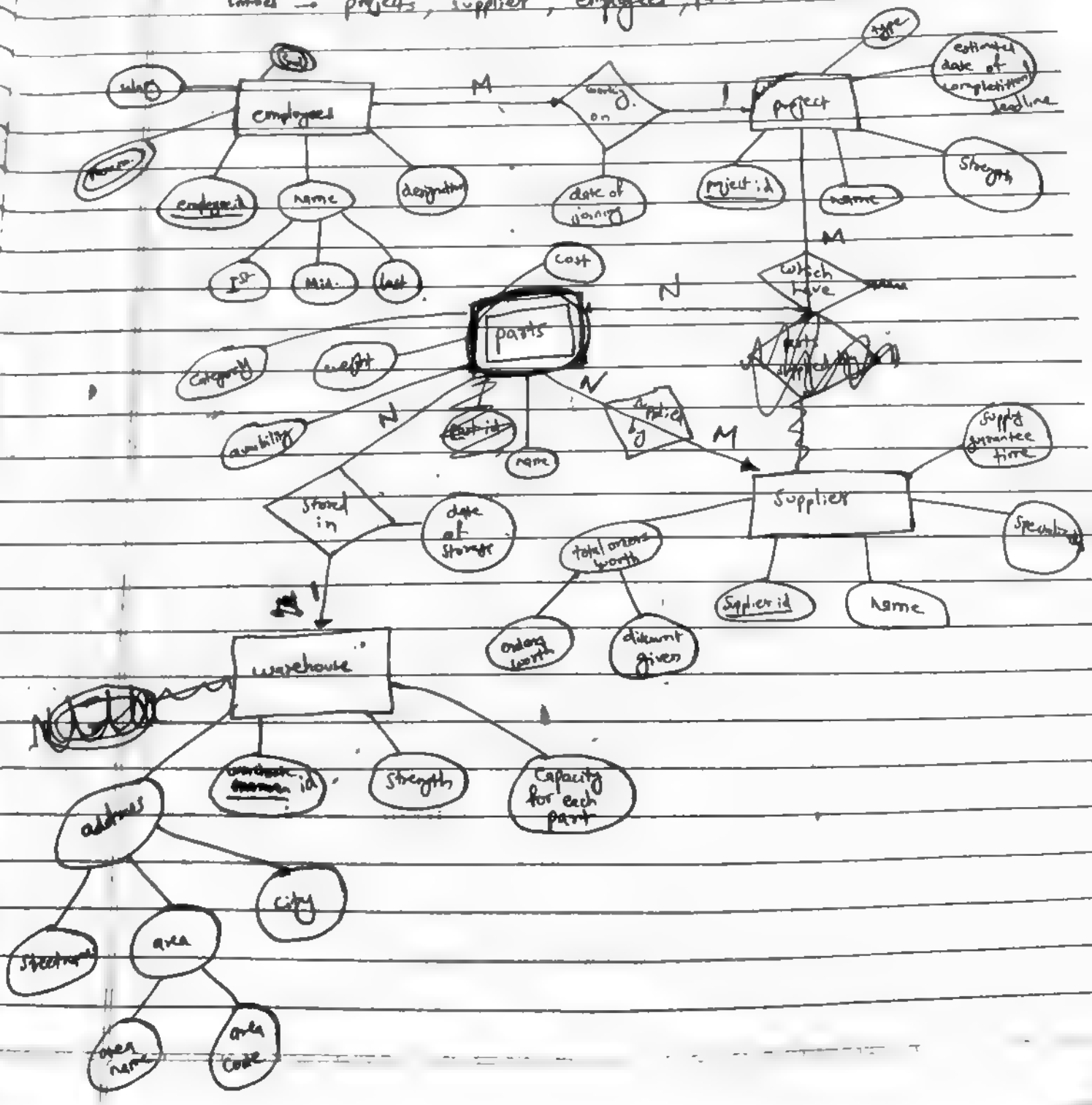
Q. Draw ER diagram showing students enrolled for the courses



M:N as ~~many~~ one student can enroll for many courses one course can have many students enrolled for the course.

Q. Draw ER diagram of a manufacturing company which records info. about projects. Projects have many parts which are supplied by the suppliers. Company maintains warehouse in which diff. parts are stored.
Various employees are working on the project.

Entities → projects, supplier, employees, parts, warehouse



EER diagram

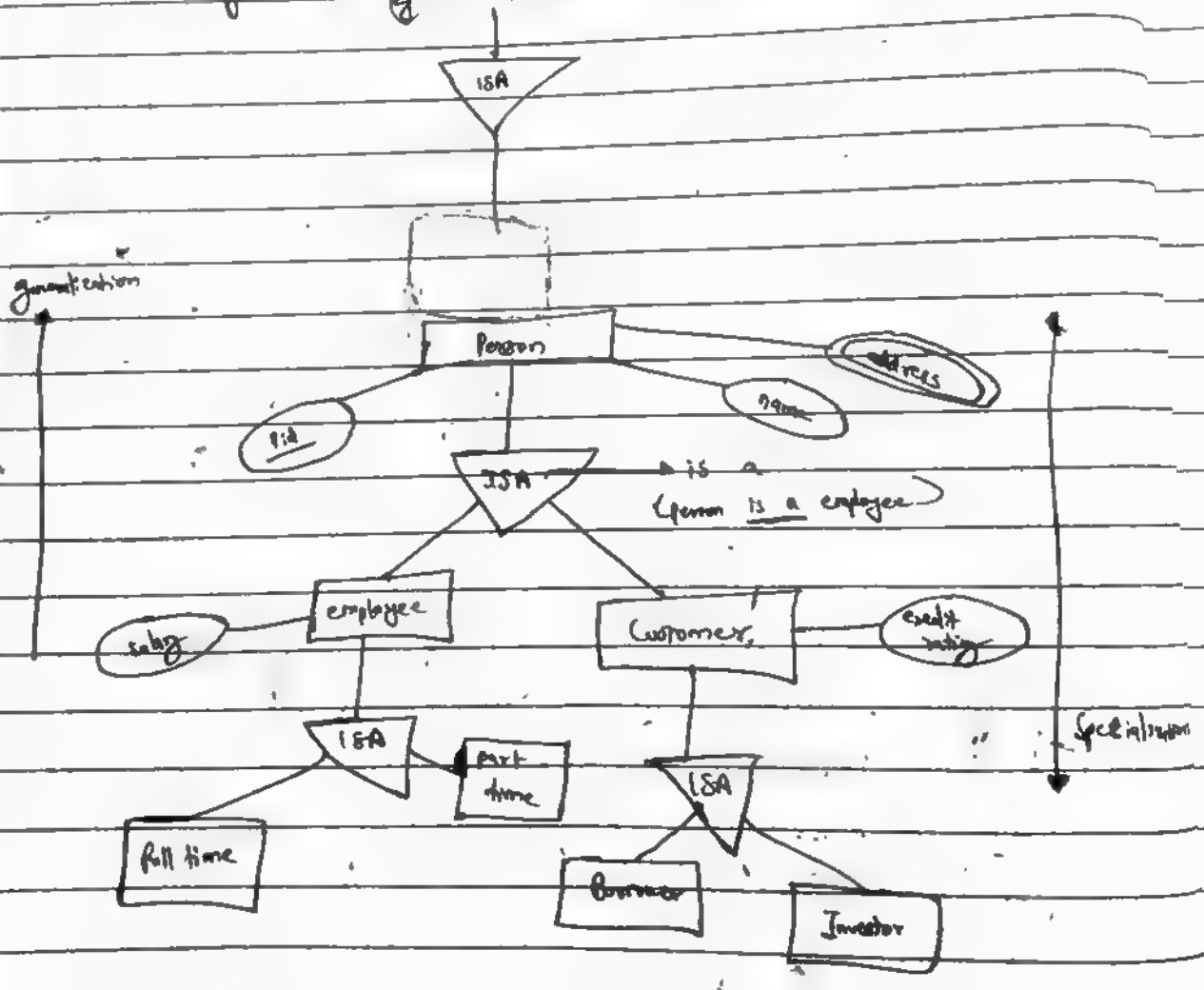
enhanced or extended version of ER diagram

It provides more functionality than the ER diagram.

① Specialization → in this higher level entity sets are divided into lower level entity sets. (Top down)

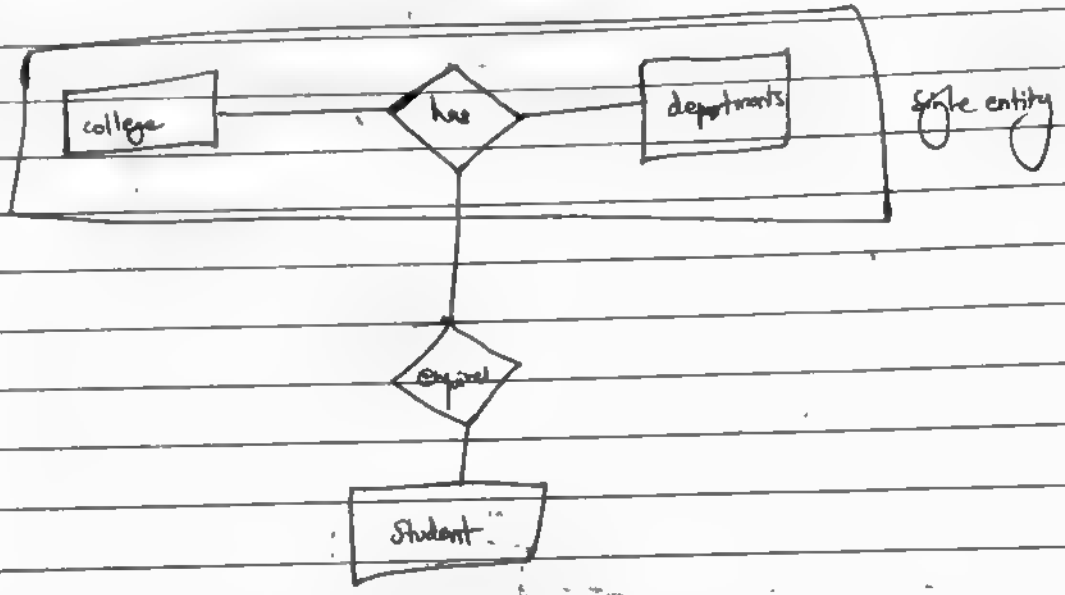
② Generalization → in this low level entity sets are merged into higher level entity sets. (Bottom up)

Symbol for both specialization and generalization :-



③ Inheritance ⟶ is the process in which all low level entities (sub-class) will inherit properties of higher level entity (super class).

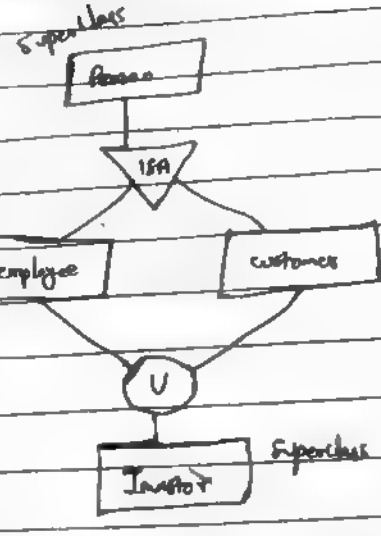
④ Aggregation ⟶ is modelling technique in which rel^n between two entities is treated as single entity.



In this diagram, rel^n between college and department acting as a single entity which is in rel^n with another entity called student. If student visits a college, he will enquire about both college as well as the department.

⑤ Categorization ⟶ is process in which subclass will have more than one superclass.

continued on next page

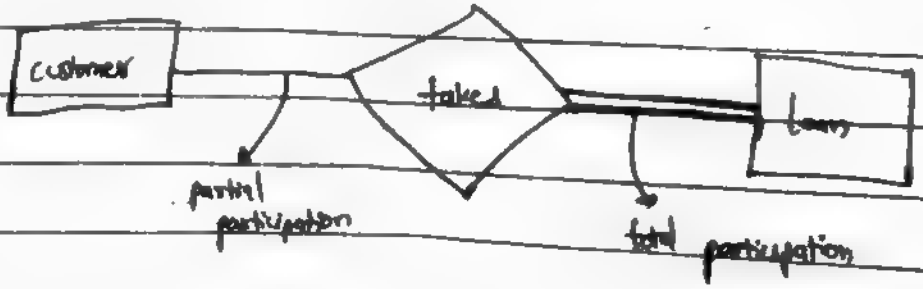


In this employee is a subclass who is a member of two superclasses:- person as well as investor. Represented with symbol → 

Participation Constraint :

① Total participation → If every entity from entity set E_1 is participating in rel" set R, then it is termed as total participation.

② Partial participation → If some of the entities are exempted are participating in rel" set R, then it is termed as partial participation.



Bank is not an entity as it is an organization

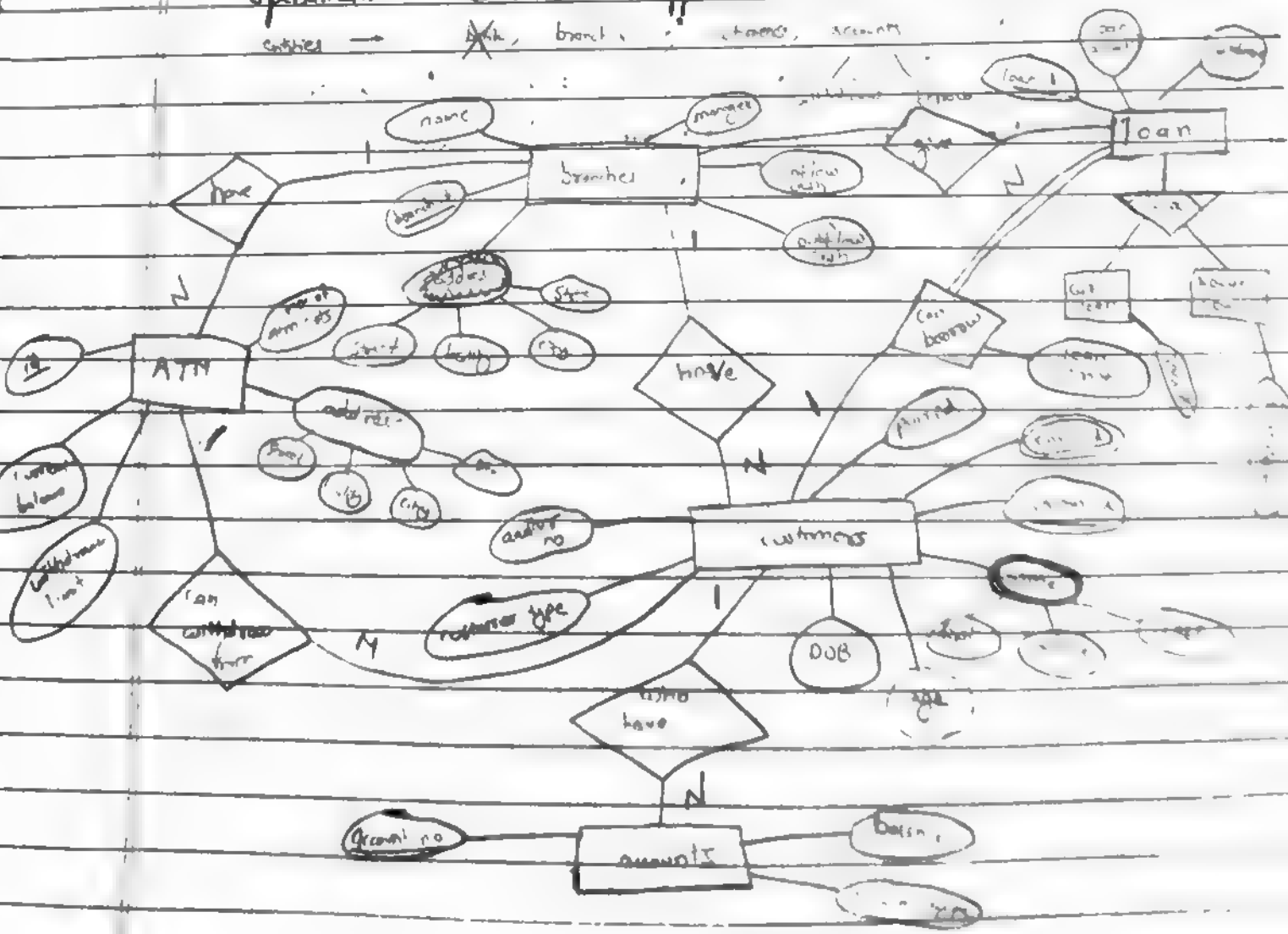
Not all customers are taking loan from a bank and therefore we have partial participation (for customer)
For each loan given we have customer id associated. Therefore we have total participation (for loan)

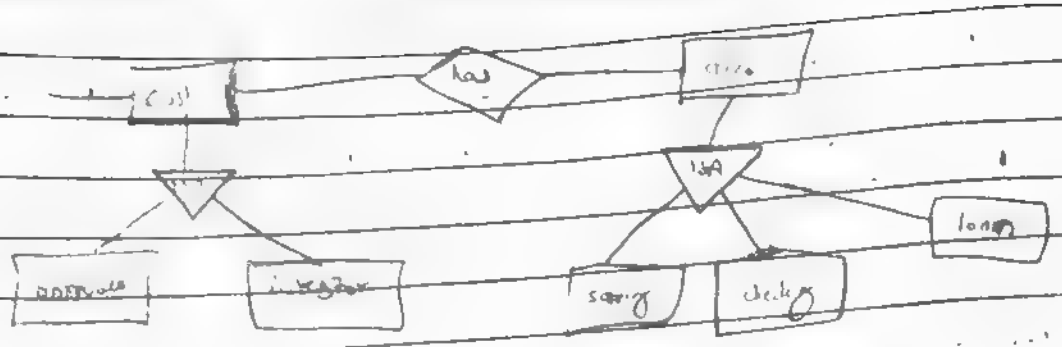
FER examples

1. Bank has many branches with many customers
2. A customer can open different kinds of accounts with the bank.
3. Any customer of the bank can take loan from the bank.
4. Banks have also installed ATM machines from which customer can withdraw money.

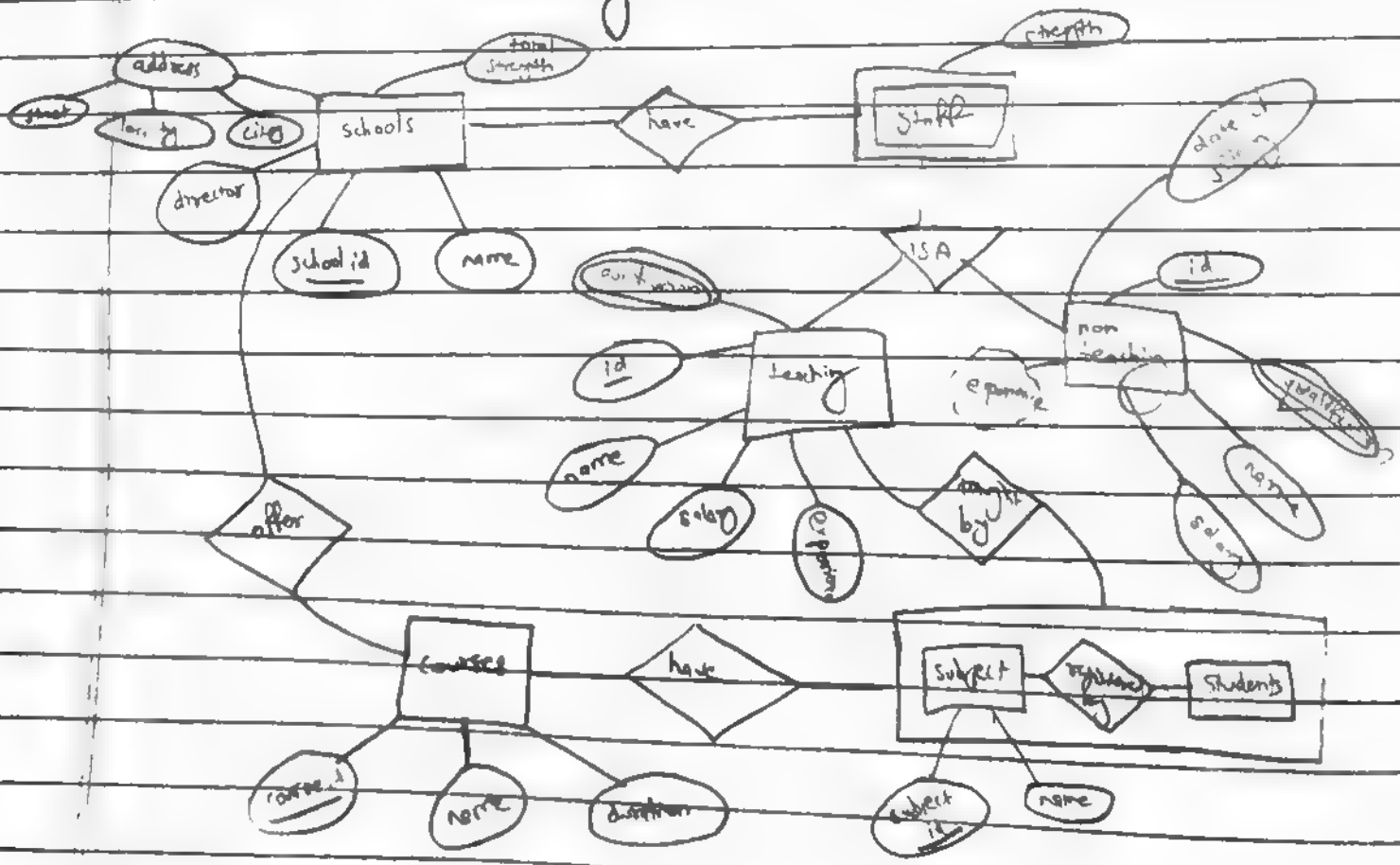
Draw EER diagram and use aggregation, generalization or specialization where ever applicable.

entities → ~~Bank~~, branch, customer, account





Q. University has many academic units called schools.
Each school is headed by a director of school.
School has teaching and non teaching staff.
School offers many courses.
A course consists of many subjects.
A subject is taught to the student who have registered for the subject by a teacher.
Draw a ER diagram



Mapping of ER/EER model into relational model

Rules for mapping

① Entity

(a) Strong → create separate table

(b) weak → separate table

(Primary key of main table + Any col.ⁿ key of weak entity)

② Attribute

(a) Composite attr. → no separate table

(b) multivalued → create separate table

③ Relationship

(a) 1:1 → No separate table to be created.

(b) 1:N → No (1's primary key added in 'M's table)

(c) M:1 → No

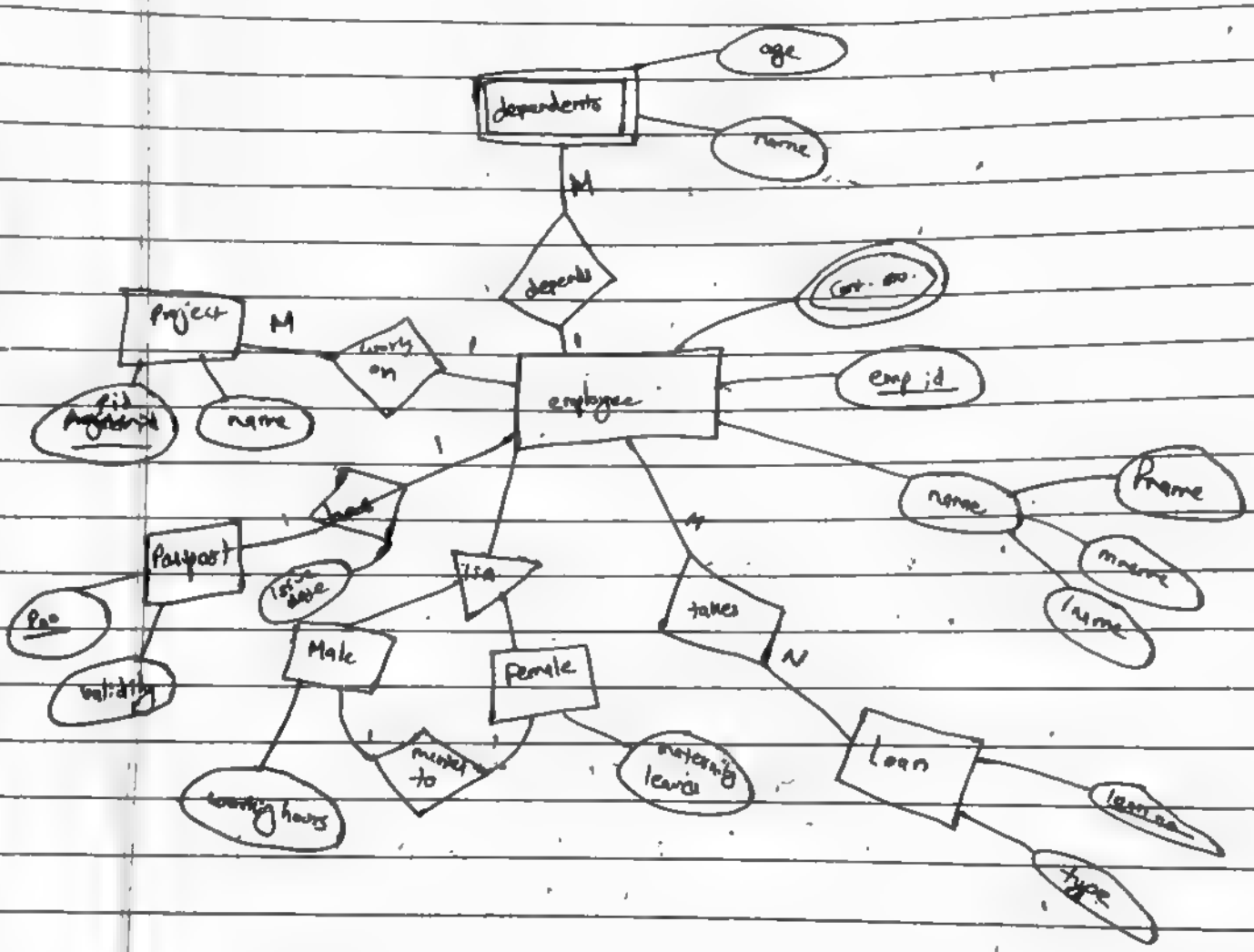
(d) M:N → Yes

④ Inheritance

Each sub class will have primary key of parent class

⑤ If descriptive attr. is present, create separate table.

Descriptive attr. → attr. for rel^n



1. emp (empid, fname, mname, lname)

2. contact (empid, contact no)

3. project (pid, name, empid)

4. male (Mempid, working hours, fempid)

5. female (fempid, maternity leaves)

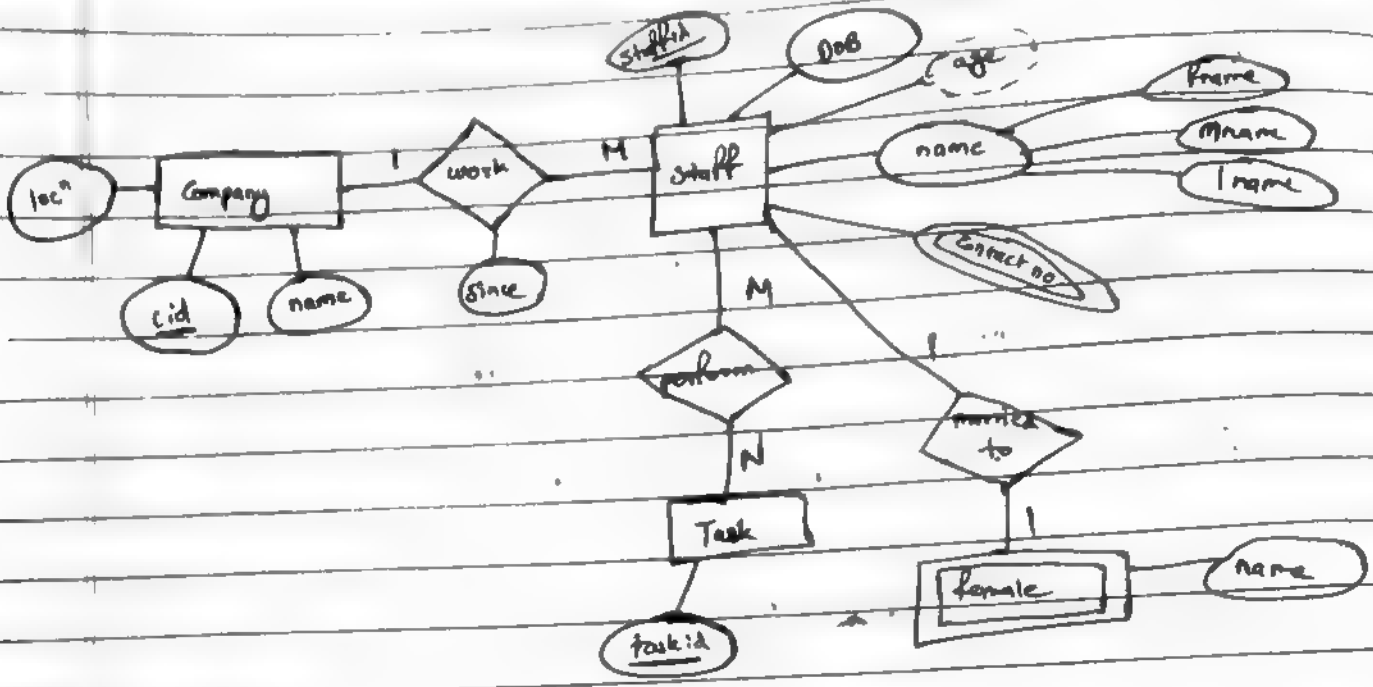
6. Dependencies (empid, name, age)

7. passport (pno, validity)

8. has (pno, empid, issue date)

9. loan (loanno., type)

10. takes (empid, loanno.)



Company (cid , loc" , name)

work (cid , staffid , since)

interesting! No need to create col^n here for age

staff (staffid , DoB , ~~age~~ , fname , Mname , lname , ~~cid~~)

Contact no (staffid , contact no.)

it will not come here as 'work' table has Cid attribute

task (taskid)

perform (taskid , staffid)

female (staffid , name)

↳ Primary key of main entity + any attribute of weak entity

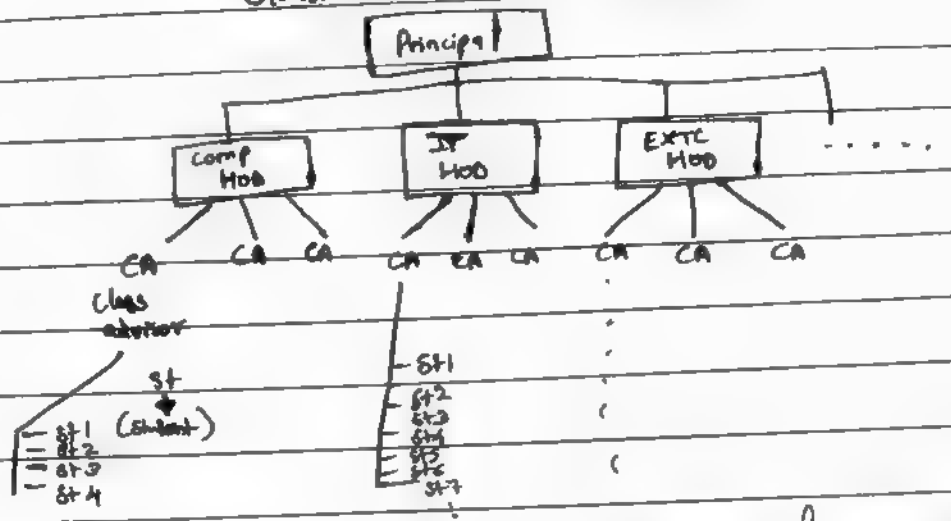
Data models in DBMS

It describes how components are structured together.

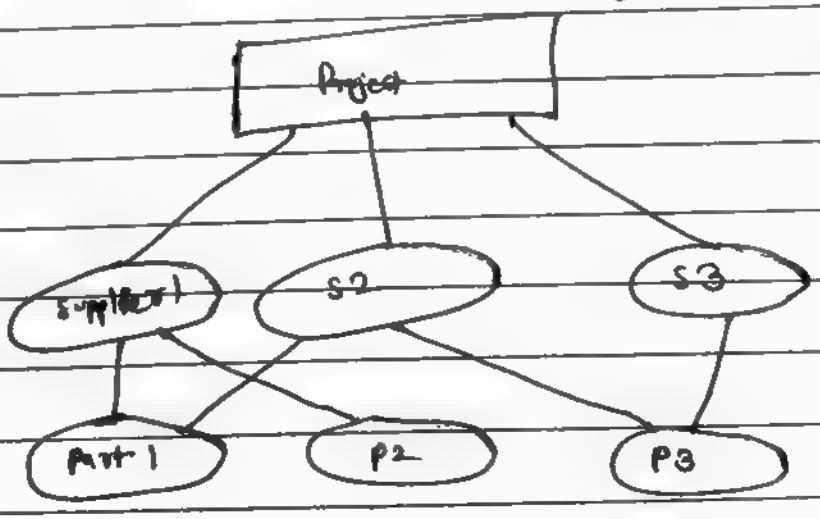
① Hierarchical model : It represents hierarchy of an organization. (level of detail)

It supports only 1:M cardinality.

Oldest model.



② Network model : It supports all the types & cardinalities.



③ ER model : It represents entities and their rel^n.

④ Relational model : If data is represented in the form of 2-D table it is rel^l model.

Relational schema

It is collection of metadata which describes conceptual / logical schema.

SYNTAX for date → dd/mm/yyyy

SQL (Structured Query language)

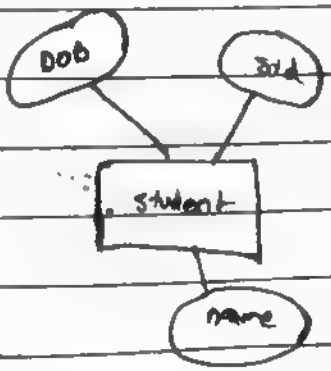
- DDL (data definition language)
  1. Create
  2. Alter
  3. DROP
  4. Rename
  5. Truncate
  6. Describe
- DML (data manipulation language)
  1. Insert
  2. Update
  3. Delete
  4. Select
- DCL (data control language)
  1. Grant
  2. Revoke
- TCL (transaction control language)
  1. Commit
  2. Save point
  3. Rollback

Data definition language

1. Create : It is used to create new table for DB.

SYNTAX : Create table tablename (
columnName1 datatype (size),
columnName2 datatype (size),
.....);

eg:- create table Student (
sid int (9),
name Varchar (20),
DoB date);



9 digits

no need to mention size

② Alter: It allows users to modify the structure of DB.

SYNTAX: alter table tablename
add columnname datatype(size);

eg:- alter table Student add marks int(8);

sid	name	DOB	marks

For multiple col^n's :-

alter table tablename
add columnname1 datatype(size),
add columnName2 datatype(size);

For altering after a particular col^n and not at the end :-

alter table tablename
add ColumnName datatype(size)
after ColumnName;

sid	name	DOB	marks	contactno

eg:-
alter table Student
add contactno int(10)
after name;

For altering at the first position of table :—

```
alter table tablename
    add columnName datatype(size)
    first;
```

eg:- alter table student
add emailId varchar(20)
first;

To modify datatype of a column :—

```
alter table tablename
    modify column
        columnName datatype(size);
```

eg:- alter table tablename
modify column
sid varchar(5);

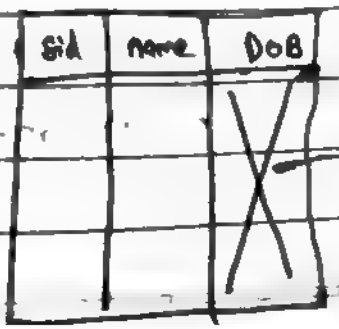
To change sid from 513 to B-513

To drop a column from DB :—

```
alter table tablename
    drop column
        columnName;
```

```
alter table tablename
    drop column
        columnName1,
    drop column
        columnName2;
```

alter table student-
drop column
DoB;



③ Rename → It allows you to rename the table name or columnName.

SYNTAX :- rename table
oldname to newname;

eg:- rename table
student to student_info;

↓

will change the table name to 'student_info'.

To change column name :-
SYNTAX :- alter table tablename
change column
oldvalue newvalue datatype(size);

[there is no 'to' here.]

eg:- alter table student_info
change column
old rollno;

④ Drop → used to drop the entire table

SYNTAX :- drop table tablename

eg:- drop table student_info;

↓

In this attributes as well as records are deleted.

(5) Truncate :- It delete all the records from the table but attribute names will be retained.

SYNTAX: truncate table tablename;

eg:- truncate table student_info;

(6) Describe :- It describes the structure of the table.

SYNTAX: desc tablename;

eg:- desc student_info;

(will describe the structure along with columnName, its datatype and size)

DML (Data manipulation language)

1. insert
2. update
3. delete
4. select

(1) Insert → used to insert no. of rows into the table

SYNTAX:- insert into tablename
values (value1, value2, value3. .....);

insert into student values (1, 'Parth', '2000-1-1');

id	int(9)
name	varchar(16)
DOB	date

insert into student values (2, 'XYZ', NULL);
↳ no DOB found

⑦ Update :-

Update command is used to modify existing value of column.

SYNTAX: update tablename
set newvalue
where oldvalue;

eg:- update student set id=3 where
name = 'XYZ'
(Change id=3 where name = 'XYZ')

[if multiple values have the same value, both the places, value will be changed]

⑧ Delete :- Delete command is used to delete no. of rows from the table.

SYNTAX: delete from tablename
where condition;

eg:- delete from student
where id=3;
(will delete rows where id=3)

④ select :- Select command is used to select no. of rows from table.

SYNTAX: select column name (s) from table name
(* → all the column names are selected)
where condition;

select * from prof_details /* All columns are selected */
where sal = 60000;

(id, name, sal)

select id from prof_details /* only id is displayed */
where sal = 6000;

(id)

id	name	sal
1	XYZ	60000
2	PQR	60000
3	ABC	65000

DBMS constraints

- Domain integrity constraints
  - ① check
  - ② default
  - ③ NOT NULL
- Entity integrity constraints
  - ① unique key
  - ② Primary key
- Referential integrity constraints
  - ① Foreign key

Constraints :- Before entering values in the table, values are checked against the constraints.
If constraints are satisfied, then only values are inserted in a table.

for data integrity

① Check constraint

(i) SYNTAX :- create table tablename (
col1 datatype (size),
col2 datatype (size),
check (condition));

eg:- create table Students (
name varchar (16),
age int (2),
check (age > 18));

(this becomes the default constraint name)

insert into table Students ✗
values ('ABC', 16); (will not be added in table)

insert into table students ✓ (will be added in table)
values ('PQR', 22);

(ii) To add a check condition after creating table
SYNTAX :- alter table tablename
add constraint Constraint_name
check (condition);

eg:- alter table students
add constraint chk1
check (age >18);

(iii) To drop constraint from table

SYNTAX: alter table tablename
drop constraint constraint_name;

(2.) default constraint

default constraint is cred when user don't know value for a specified column.

(i) creating table using 'default' :-

SYNTAX: create table tablename (
c1 datatype (size),
c2 datatype (ssze),
c3 datatype (size) default 'value');

[Insert into student values (. . . . .)
insert into table values (. . . ; → without the default corresponding val. specified

(ii) Change default after creating table

SYNTAX: alter table tablename. alter columnname set default 'value';

alter table tablename
alter 'columnname'
set default 'value'

(iii) drop a default value

SYNTAX: alter table tablename
alter column columnname
drop default;

→ Unique can have duplicate values but null values is not accepted.

PAGE NO.
DATE / /

③ Not NULL constraint :-

SYNTAX :- create table tablename (
c1 datatype(size),
c2 datatype(size),
c3 datatype(size) not null);

(iii) after table created :-

SYNTAX :- alter table tablename
modify column columnname datatype(size)
not null;

→ create table student (
sid int(9),
name varchar(25) not null,
city varchar(50));

insert into student values (3, 'ABC'); ✓
(3, NULL, 'MUM'); ✗
(2, 'ABC', NULL); ✓
(2, 'ABC', 'MUM'); ✓

(iv) drop 'not null' :-

alter table tablename
modify column columnname datatype(size)
null;

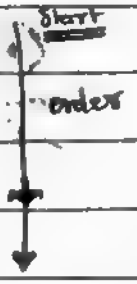
PAGE No.
DATE / /

Entity integrity Constraint

(i) Unique constraint → duplicate values are not supported but we can have null values.

SYNTAX:- create table tablename (
c1 datatype (size),
c2 datatype (size) unique,
c3 datatype (size));

(1, 'P', 'Mum') ✓
(2, 'Q', 'Mum') ✓
(3, 'P', 'Banglore') X
(4, NULL, 'Banglore') ✓



(i) After table created

SYNTAX: alter table tablename
add constraint constraintname
unique (columnName);

(ii) drop a unique constraint.

SYNTAX:- alter table tablename
drop index constraintname;

③ Primary key :- it is the combination of unique + not null

SYNTAX :- create table tablename (
c1 datatype (size), primary key
c2 datatype (size),
c3 datatype (size));

insert
(1, 'A', 'Mum');
(2, 'B', 'Bangalore');
(1, 'C', 'DEF'); X
(NULL, 'D', 'EFG'); X
(3, NULL, NULL);
order of execution

(i) after table created

alter table tablename
add constraint constraintname
primary key (columnname);

(ii) drop 'primary key' constraint from a constraint-name/ column

alter table tablename
drop primary key;

Referential integrity constraint :- It focuses on primary - foreign key relationship.

Sid	name	age	dept id
1	A	18	10
2	B	19	20
3	C	19	10
4	D	21	20
5	E	20	30

student

dept id	branch
10	IT
20	COMP
30	EXTC

department

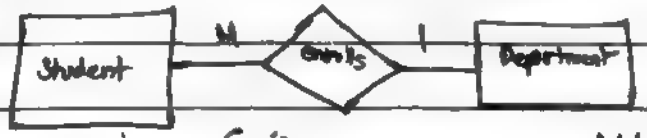
Sid → primary key for 'student' table

dept id → " for 'department' table

also → acts as a normal col^n in student table, referred as foreign key.

→ foreign key will have NULL as well as duplicate values.

(6, F, 21, 40) X (as 40 is not there in department table)



(1's primary key in M's table)

SYNTAX :- create table tablename (

c1 datatype (size),

:

cn datatype (size),

constraint constraintName foreign key (columnName)

references anothertablename (columnName));

eg:- create table student (

constraint fk1 foreign key (deptid)

references dept (deptid));

PAGE NO.
DATE / /

(i) after table created

SYNTAX:- alter table tablename
add constraint constraintname foreign key (columnname)
references anothertablename (columnname);

(ii) drop foreign key from a constraint name

(2) alter table tablename
drop index constraintname;

(1) alter table tablename drop foreign key constraintname;

space

Transaction control language

It is used in client-server architecture

Transaction → series of statements used in client-server architecture.

1. Commit
2. Rollback
3. Savepoint

Commit → used to save changes permanently in the DB.
once changes are made, changes cannot be undone.
SYNTAX :- Commit;

rollback → used to restore the old values upto last commited step.
SYNTAX : rollback;

savepoint → used to save changes temporarily in DB.

Creating of savepoint :-
SYNTAX :- Savepoint savepoint_name;

roll back with savepoint :-
SYNTAX :- rollback to savepoint_name;

eg:-

id	name	sal
1	Ajay	9000
2	Vijay	5000
3	Sanjay	6000

prof

1. begin;
2. update prof set sal = sal + 1000 where id = 1;
3. select * from prof; (will show ajay → 10000)
4. rollback;
5. select * from prof; (will show ajay → 9000)
6. Same as 2.
7. Commit; (changes made permanent)
8. select * from prof; (10k)
9. rollback;
10. select * from prof; (still shows 10k)

interesting commit indicates client-server arch. has completed so use use begin command again

(as ...)

(11) begin;
(12) same (2) (11k)
(13) same as (2) (12k)
(14) rollback;
(15) select * from prof (10k)

(16) start transaction; (to use Savepoints)

(17) savepoint sp1;
(18) delete from prof where id=1; (ajay X)
(19) select * from prof;
(20) savepoint sp2;
(21) delete from prof where id=2; (vijay X)
(22) select * from prof;
(23) savepoint sp3;
(24) delete from prof where id=3; (sanjay X)
(25) same as (22)
(26) rollback to sp2; (Sanjay, vijay added)
(27) same as (22)

savepoint 2 [by default savepoint 3 is performed]

SQL operators → Operator is a special word used in databases for describing operations. There are 6 types of operators in SQL:-

1. Arithmetic
2. Comparison
3. Logic
4. SQL operators
5. Range searching
6. Pattern matching

1. Arithmetic operators

$x = 20$
$y = 10$

operator	description	eg
+ (addition)	adds value in either side of operator.	$x + y = 30$
− (subtraction)	Subtracts right hand operand from left hand operand.	$x - y = 10$
* (mult.)	multiplies value in either side of operator.	$x * y = 200$
/ (divide)	left hand operand gets divided by right hand operand	$x / y = 2$

what happen when we interchange select id, sal ... instead of select sal, id ... check?

Examples of arithmetic operators

emp_table

empid	fname	lname	city	sal
1	Ajay	Sharma	Mumbai	5000
2	Vijay	Mehta	Pune	10000
3	Sanjay	Patil	Mumbai	7000
4	Jayesh	Sharma	Pune	2000

Display sal, empid of the employee by adding 1000 to it.

```
select id, sal as old_sal,
sal + 1000 as new_sal from emp_table;
```

id	old_sal	new_sal
1	5000	6000
2	10000	11000
3	7000	8000
4	2000	3000

PAGE No.
DATE / /

Display first name and sal, by dividing them 2.

select fname, sal as old_sal,
sal/2 as new_sal from emp_table;

fname	old_sal	new_sal
Ajay	5000	2500
Vijay	10000	5000
Sanjay	7000	3500
Jagesh	2000	1000

Display all the details of the employees by giving 5% raise in sal.

Select empid, fname, lname, city, sal as old_sal,
sal+(sal*5)/100 as new_sal from emp_table.

empid	fname	lname	city	sal
1	Ajay	Sharma	Mumbai	5250
2	Vijay	Mehta	Pune	10500
3	Sanjay	Patil	Mumbai	7350
4	Jagesh	Sharma	Pune	2100

DATE / /

Comparision operator

1. = (equal)
   Checks value of two operands:-
   If equal → returns true

2. != (not equal to)
   Checks value of two operands:-
   Not equal → returns true
   Equal → returns false

3. < (less than)
   Checks if value of left operand is less than value of right operand.
   If yes → returns true

4. <= (less than, or equal to)

5. > (greater than)
   Value of left operand is greater than value of right operand.
   If yes → true

>= (greater than or equal to)

1. emp whose sal is 7000.
2. display all the employees whose sal >= 5000 7000.
3. display record of all the employee by adding 2000 to the salary only when sal < 7000.

PAGE NO.
DATE / /

⑥ select * from emp_table where sal = 7000.

⑦ select * from emp_table where sal >= 7000.

⑧ select empid, fname, lname, city, sal as old_sal, sal + 2000 as new_sal where sal < 7000 from emp_table where sal < 7000;

3 Sanjay Patil Mumbai 7000

2 Vijay Mehta Pune 10000

3 Sanjay Patil Mumbai 7000

empid, fname, lname, city, old_sal, new_sal

Logical operators

AND :- it returns True if all the conditions are satisfied.

OR :- it returns True if one of the condition is satisfied

Q. Display the information of all the employees with first name as 'sanjay' and last name as 'patil'.

select * from emp_table where fname = 'sanjay' and lname = 'patil';

empid = 3 ← output

For or → 3, 5, 6

emp_id	fname	lname	city	sal
1	ajay	sharma	mumbai	5000
2	vijay	Mehta	pune	10000
3	sanjay	Patil	mumbai	7000
4	jayesh	Sharma	pune	2000
5	mayuresh	patil	kolkata	15000
6	ajay	patil	mumbai	7500

fname lname

Q. 'sanjay' or 'patil'

select * from emp_table where fname = 'sanjay' or lname = 'patil';

output →

3 —

5 —

6 —

Q. fname = 'ajay' or lname = 'sharma'

emp id	fname	lname	city	sal
1	ajay	sharma	mumbai	5000
3	sanjay	patil	mumbai	7000
4	jagesh	sharma	pune	2000

Q. List all employees having salary < 7000 and city as mumbai.

select * from employees where sal <=7000 and city = 'mumbai';

1. ajay sharma mumbai 5000
3. sanjay patil mumbai 7000

Q. Display information of all employees having fname as 'ajay' or 'jagesh' and last name as sharma

Select * from employees where (fname = 'ajay' or fname = 'jagesh') and lname = 'sharma';

1 ajay sharma mumbai 5000
4 jagesh sharma pune 2000

PAGE NO.
DATE

Range searching operators

BETWEEN / NOT BETWEEN
IN / NOT IN

Two types of operators are included in range searching:
It is used to select the records from the table when we know the exact value

IN:- SYNTAX :- select columnnames (*) from tablename where columnname in (value1, value2, ... valuen);

id	fname	lname	sal	city
1	ajj	sharma	5000	mum
2	vijay	mehta	6000	pune
3	sanjay	patil	7000	mum
4	yogesh	sharma	8000	pune

① select * from emptable where lname in ('sharma', 'patil');

1, 3, 4

② Select * from emptable where (lname = 'sharma' and lname = 'patil');

{ } empty set

for OR → same as ①

neither lname nor equal to 'sharma' nor 'patil'
AND

Select * from empt table where (lname != 'sharma' and lname != 'patil');

2

substitute is not in

2. vijay mehta 1000 pune

not equal to 'sharma' or 'patil'

↳ or → select * from emptable where (lname != 'sharma' or lname != 'patil');

1, 2, 3, 4

BETWEEN

- used to select values based on ranges

SYNTAX: Select column_name (*) from tablename where column_name between min_value and max_value;

① Select * from emp_id where sal between 2000 and 7000;

1, 3, 4

NOT BETWEEN

① select * from emp_id where sal < 2000 and sal > 7000

↳ { } empty set

② select * from emp_id where sal < 2000 or sal > 7000

↳ Not between

2

PAGE NO.
DATE / /

④ select * from emp where fname between 'Ajay' and 'Sanjay';

1, 3, 4

eg fname >= 'Ajay' and fname <= 'Sanjay'

⑤ select * from emp where fname between 'A' and 'S';

1, 4

very interesting eg 'Sanjay' > 'S',
so we will discard Sanjay,
only 'Ajay' and 'Jyoth'

Set operators :- used to combine result set of

cid	cname	ccity	ccountry
1	Priyanka	Mum	Ind
2	Vijay Vipul	Delhi	Ind
3	Vijay	Berlin	Germany

Customer

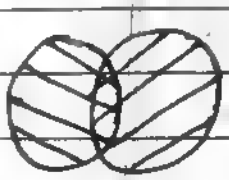
eid	ename	ecity	ecountry
1	Sachin	London	UK
2	Vijay	Berlin	Germany
3	Sanjay	NY	USA
4	Rohit	Texas	USA
5	Vijay	Delhi	Ind

employee

used to combine output of multiple select statements. Four types:-

- union all
- union
- intersect
- minus

Union all :- It is used to combine two select statements together with keeping duplicate records.



SYNTAX :- select col from ontable
union all
select col2 from anothertable;

eg:- select cid from customer
union all
select eid from employee

cid
1
2
3
1
2
3
4
5

select cid from customer
union all
select eid, ename from employee

limitation: In set operator ① we should have equal no. of columns and ② their datatype should be matched.

union:- It is variation of union all in which duplicate values are removed and only distinct values are displayed.

eg:- select cid from customer
union
select eid from employee;

cid
1
2
3
4
5

cname, city

Select cid from customer
union
select ename, city from employee

cname	city / country
Priyanka	Mum
Vipul	Delhi
Vijay	Berlin
Kalpana	london
Sanjay	NY
Vashid	Texas
Vijay	Delhi

in this case we compare (Vijay, berlin) with (Vijay, delhi)

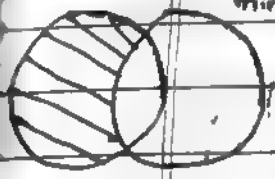
DATE / /

• intersect :- it selects common records from both the tables.

select cname, ccity from customer
intersect
select ename, ecity from employee

cname	ccity
Vijay	Berlin

minus :- it will select uncommon records of first select statement.



select cname, ccity from customer
minus
select ename, ecity from employee

cname	ccity
Priyanka	Mumbai
Vipul	Delhi

Then Table page

③ select all employee where fname 'ja' in it.

id	fname	lname	sal
1	Vijay	Mehta	5000
2	Ajay	Chavan	3000
3	Sanjay	Patil	4000
4	Jagesh	Sharma	2000

```
Select * from emp
where fname like '%ja%';
```

④ select all employees where second & last name is 'a'.

```
select * from emp
where fname like '_a%';
```

→ Sanjay, Jagesh

⑤ select all employees where for fname we have exactly six characters.

```
select * from emp
where fname like '------';
```

→ Sanjay, Jagesh

⑥ find all employees where fname should contain 'ja' but does not start with 'ja'.

⑦ Select all the employees in which for fname 'a' is in second position and fname must end with 'y' or 'h'

```
select * from emp
where fname like '_a%ja%' and
fname not like 'ja%';
```

redundant

o/p → 123

```
select * from emp
where fname like '_a%y' or fname like '_a%h';
```

most efficient

```
select * from emp
where fname like '_%ja%';
```

o/p → y, h

(Sundaram)

first this page

Pattern matching operators

id	fname	lname	sal
1	Vijay	Mehta	10000
2	Ajay	Sharma	5000
3	Sanjay	Patil	7000
4	Jayesh	Sharma	2000

emp.

① select all the employees where fname ends with y.

In pattern matching we have:—

- like / not like
- is null / is not null

like: SYNTAX: select columnNames (*) from tablename where columnName like 'condition';

wildcards

_ → exactly one value

% → 0 or more characters

```
Select * from emp
where fname like '%y';
```

Vijay
Ajay
Sanjay

② select all employees where last name starts with 'S'

```
select * from emp
where lname like 'S%';
```

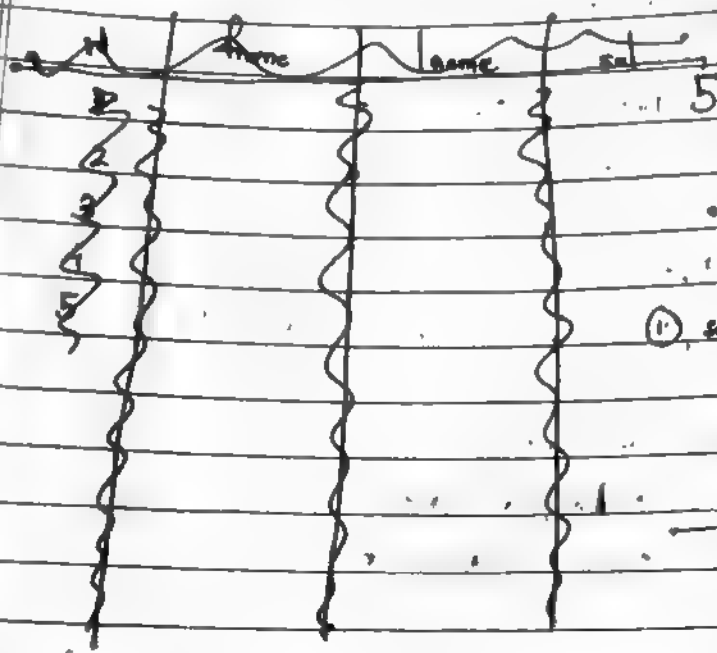
Ajay Sharma
Jayesh Sharma

PAGE NO.
DATE / /

is null / is not null

↓

when a Null / Hole is not null



We are adding val.

5	Satish	Bhatt	NULL

to the existing table

① select * from emp where sal is null;

id	fname	lname	sal
5	Satish	Bhatt	NULL

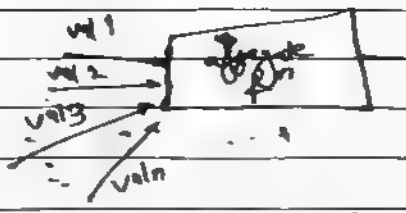
② select * from emp where sal is not null

↳ 1 to 4

grouping, ordering clause and agg. more fns

Aggregate functions

id	fname	lname	sal
1	Ramesh	Kale	6400
2	Vijay	Sharma	9400
3	Sujay	Patil	67000
4	Mukesh	Patil	4200
5	Ajay	Bhoir	10000
6	Mayuri	Bhosale	15600
7	Pramod	Patil	NULL
8	NULL	NULL	NULL



SYNTAX :- select aggregate_fname (columnName) from tablename where condition;

Aggregate fn's will take set of values as an input and produce only a single value as an output.

SUM (clname) → provides total of selected column excluding NULL values.

AVG (colname) → gives avg. excluding NULL

MIN (colname) → displays minimum value of a specified col.

MAX (colname) → displays max value for a specified col.

COUNT (colname) → gives total no. of values of a particular column excluding NULL.

PAGE NO.
DATE / /

ignores NULL val

COUNT(*) → displays no. of records, a particular table has.

Q: find total sal. of employee :—

```
select sum(sal) from emp;
```

sum(sal)
112009

```
Select sum(sal) as Total_salary from emp;
```

SUM will exclude entire row if for sum(sal+id) either sal or id is NULL.

sum 7, 8 excluded as salary is NULL

```
Select sum(sal+id) from emp;
```

sum(sal+id)
112021 → (1 to 6)

Q. avg salary of employees

```
Select AVG(salary) from emp;
```

avg(sal)
18666.67

Q. find avg salary of employees whose salary is greater than 10000.

```
Select AVG(salary) from emp where sal > 12000;
```

avg(sal)
14100.

Q. find highest salary of employee
select MAX(sal) from emp; → 6700

Q. find smallest salary from of employee
Select min(sal) from emp; → 4200

Q. display total no. of records employee table has
select count(*) from emp; → 8

Q. select count(sal) from emp;
→ 6 (because count(column_name) does not count NULL)

Q. no. of employees whose name starts with 'p'
↓
Select count(ename) from emp where ename like 'p%';

Q. no. of NULL records 'sal' column has
✗ select count(sal) from emp where sal is null; (does not run in MYSQL)
✗ Select count(sal) from emp where sal = Null; (does not run in MYSQL)
✓ Select count(sal) from emp where sal not like '_%';
✓ select count(*) - count(sal) from emp; → 2

Q. total no. of values, total null values, not null values for sal column.

select count(*), count(*) - count(sal), count(sal) from emp;

count(*)	count(*) - count(sal)	count(sal)
8	2	6

Not NULL values in table

Group by

set of values are combined together and on these set of values aggregate fn is applied.

SYNTAX :- select columnName,
aggregate_fn_name (columnName)
from tablename where
condition group by
columnName1, columnName2;

display total salary of employee according to the department

select deptname, sum(sal) from emp group by deptname;

because according to deptname we have to disp. salary

Table (in the behind table we add deptname)

id	deptname
1	Sales
2	acc
3	sales
4	HR
5	marketing
6	acc
7	HR
8	HR
9	marketing

dept	sum(sal)
acc	24400
HR	4200
marketing	10000
sales	73400

display total salary of employee where sal > 9000 according to deptname

1000	sales

select deptname, sum(sal) from emp where sal > 9000 group by deptname;

dept.	sum(sal)
acc	24400
marketing	10000
sales	67000

Q. select total salary of employee according to deptname and fname.

select fname, deptname, sum(sal) from emp group by deptname, fname;

exp 6. To study and implement different types of databases in

b513 PAGE NO. DATE / /

mysql -u root -p

-u → username

-p → password

show databases; → for displaying how many databases currently in the system.

create database b513;

use b513 → use our db

show tables; → to show our list of tables

exp.4 To study and implement types of constraints in MYSQL

exp.5 To study and implement TCL commands

deptname	Ename	sum(sal)
acc	manoj	15000
acc	vijay	9400
HR	null	null
HR	mukesh	4200
HR	pramod	null
marketing	ajay	10000
sales	ramesh	7400 → (6400 + 1000)
sales	sanjay	6700

Q. display total salary of emp according to deptname where total sal is greater than 10000

select deptname, sum(sal) from emp where sum(sal) > 10000 group by deptname;

(NOT SUPPORTED) ✗

WHERE clause is not applicable for aggregate fn, ∴ they introduced having clause

select deptname, sum(sal) from emp

group by deptname having sum(sal) > 10000;

dept	sum(sal)
acc	24400
Sales	73400

PAGE No.
DATE / /

tone anyone
Sorrowless hearted

Aim: To study basic concepts of DBMS
- def^n of DB and DBMS
- difference between filesystem and DBMS
- arch. of DBMS
- diff. tools available in market related to DBMS
- characteristics of DBMS

- add fname and lname together
→ select concat (fname, lname) from emp;

AJAYsharma
VIJAYpatil
MUKESHpatel

add fname and lname with separator
↳ select concat_ws ('-', fname, lname) from emp;
↳→ | AJAY-Sharma |
⋮

alter table emp add column combine;
update emp set combine = concat_ws ('-', fname, lname);

→ Create a new col and insert into the column values.

Having clause:- SYNTAX:-

```
select column_name
aggregate_fn_name (columnName)
from tablename where
condition group by
columnName1, columnName2
having
aggregate_fn_name (columnName)
condition;
```

Q. display total salary of emp according to deptname where total sal is greater than 10000. and sal is greater than 9000.

dept.	sum(sal)
acc	24400
sales	67000

```
select deptname, sum(sal) from emp where sal>9000 group by deptname
having sum(sal) > 10000
```

String manipulation

① upper(cn) – converts string into uppercase
② lower(cn) – ——— " ——— lowercase
③ length(cn) – returns length of string
④ replace(cn, 'old value', 'new value') – replaces old value with a new value in a string
⑤ concat(first_col_name, second_col_name) – add two strings together
⑥ concat_ws('separator', fcn, scn) – add two strings with separator

emp_id	fname	lname	sal	city
1	AJAY	SHARMA	2000	Pune
2	VIJAY	PATIL	8000	Mum
3	MUKESH	PATEL	2500	kol

Q. display fname of emp in small letters and lname of emp in capital letters

```
select upper(fname), lower(lname)
from emp;
```

Q. replace each 'A' of first name by 'AA'

```
select replace(fname, 'A', 'AA')
from emp;
```

AAJAAY
VIJAAY
MUKESH

Q. display length of fname column

```
select fname, length(fname) from emp;
```

fname	length(fname)
AJAY	4
VIJAY	5
MUKESH	6

nested string fns.

NOTE:- we can use lower(replace(---));

Joins in DBMS

Joins are used to combine output from two different tables.

emp

eid	fname
04	Vijay
7369	Sanjay
7499	Mukesh
7521	Pramod

orders

p.id	pname	eid
284	shirt	7369
657	top	7521
865	tshirt	7521
878	Jeans	NULL

Q. display name of the employees who have ordered products

```
select emp.fname, orders.pname from emp, orders
    where emp.eid = orders.eid ;
```

fname	pname
Sanjay	shirt
Pramod	top
Pramod	tshirt

Q. Select names of employees who have purchased shirt

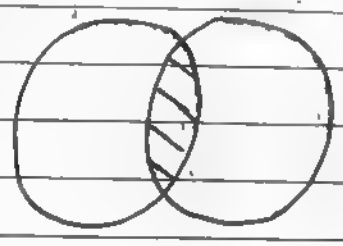
```
select emp.fname, from emp, orders where emp.eid = orders.eid
    and orders.pname = 'shirt';
```

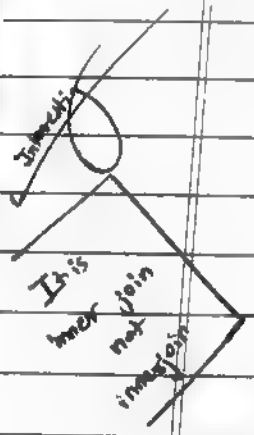
fname	pname
Sanjay	shirt

Types of joins :-

1. Inner join
2. Equi join
3. Natural join
4. left join
5. right join
6. full join
7. cross/ carterian join

(1) Inner join :- it selects rows from both tables when specified condition is specified. (CAN HAVE ANY TYPE OF OPERATOR PRESENT IN 'ON' CONDITION)





SYNTAX:

```
select cnames, from first table name
inner join second table name
on
first table.keyfield = second table.keyfield
where condition;
```

(foreign)

Q. display fname & lname of emp who have ordered products.

```
select emp.fname, orders.pname, from emp
from emp
inner join orders
products
on
emp.eid = orders
products.eid;
```

Q. with order as shirt

```
where orders.pname = 'Shirt';
```

② Equijoin:- If inner join condition contains equality operator '=', then this type of a join is equijoin.

→ eg:- this is equijoin (In Equi join, we don't have where condition.)

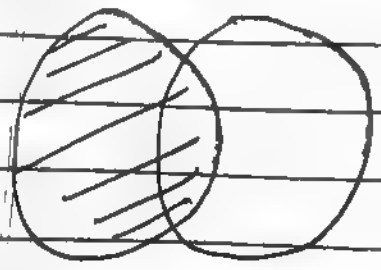
③ Natural join:- Variation of inner join, where no ON condition is present. Selection is done based on matching column. This selection is performed & automatically by DBMS.

SYNTAX:- select cnames from firsttablename
natural join second table name;

(ON ⊛ .first table.keyfield = secondtable.foreignkeyfield
if is not necessary)

④ Left join:- it selects rows from left table and if no value is present in right table, NULL is returned.

SYNTAX:- Select cnames from first table name
left join Secondtablename
on
first table.keyfield = second table.foreignkey field

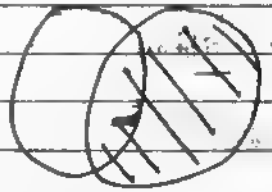


Q. $(same as below)

```
Select emp.eid, emp.fname, orders.pid, orders.pname
    from emp
        left join orders
        on
        emp.id = emp_orders.id;
```

eid	fname	pid	pname
09	Vijay	NULL	NULL
739	Sanjay	234	Shirt
749	Nikesh	NULL	NULL
758	Pramod	657	top
7581	Pramod	865	tshirt

(5) Right join :- It selects rows from right table and if no matching is found, NULL is written



SYNTAX :-

```
select cols from lefttablename
    right join secondtablename
    on
    firsttablename.keyfield =
    secondname.foreignkeyfield;
```

Q. select emp.eid, emp.fname, orders.pid, orders.pname
from emp
right join orders
on
emp.id = orders.id;

emp_eid	fname	pid	pname
7869	Sanjay	234	Shirt
7521	Pramod	657	Top
7621	Pramod	865	tshirt
NULL	NULL	878	Jeans

Full join:— It returns rows from both tables and if no match is found, NULL is returned.

SYNTAX:— select cols from firsttable_name left join secondtablename on firsttablename.keyfield = secondtablename.foreignkeyfield
union
select cols. from firsttablename right join secondtablename
on
firsttablename.keyfield = secondtablename.foreignkeyfield;

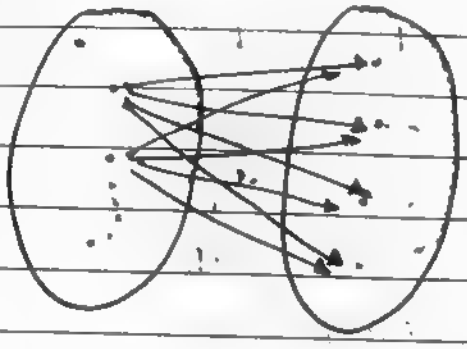
Q.

eid	fname	pid	pname
04	Vijay	NULL	NULL
7869	Sanjay	234	shirt
7199	BaMukesh	NULL	NULL
7521	Pramod	657	top
7521	Pramod	865	shirt
NULL	NULL	878	jeans

PAGE NO.
DATE / /

⑦ Cross join :- each row of left table is matched with all the values of right table.

SYNTAX: select cname from firsttablename cross join secondtable name;



eid	fname	pid	pname
04	vijj	234	shirt
04	vijj	657	top
04	vijj	865	tshirt
04	vijj	378	jeans
7369	sanjay	234	shirt
7369	sanjay	657	top
7369	sanjay	865	tshirt
7369	sanjay	378	jeans

→ Relation stands for table in DB.

interesting

Relational DB design

* Redundancy of data → some piece of information in a table will have duplicate values. Such a data is termed as redundant data.

Anomalies in DB :- (i) insertion anomaly → if we know some part of a data, and other part of a data is absent, such an information cannot be inserted in a table.

(ii) deletion anomaly → if we delete unwanted data, along with deletion of unwanted data, wanted data can also be lost.

(iii) Updation/modification → if we are going to update redundant data, change must be updated in all the places.

Normalization

Normalization is the process in which redundant data can be removed by eliminating insertion, updation and deletion anomalies.

interesting

Functional dependency

It describes constraints between two attributes. Which says that :- "for every value of X, we should have exact 1 value of Y"

Eid	Name	Salary
1	Aadi	16000
2	Manoj	16000
3	Sandip	10000
4	Vikas	10000
5	Manoj	16000

In this relation X and Y are two attributes

$X \longrightarrow Y$

In this case X is functionally determined Y if for every value of X, we have exactly one value of Y.

X → determinant

Y → functionally dependent

Duplication is allowed

Null allowed

cid → name, name → sal we have F.D

(x) (y) (x) (y)

For salary leads to name (cal → name) no BFD

↳ as 16000 → Aadi

→ Manoj

Q.

A	B	C	D
A_1	B_1	C_1	D_1
A_1	B_2	C_1	D_2
A_2	B_2	C_2	D_2
A_2	B_2	C_2	D_3
A_3	B_3	C_2	D_4

D → A ($D_2 \rightarrow A_1, A_2$)

D → B (F.D)

D → C ($D_2 \rightarrow C_1, C_2$)

A → B ($A_1 \rightarrow B_1, B_2$)

A → C (No F.D)

A → D ($A_1 \rightarrow D_1, D_2$; $A_2 \rightarrow D_2, D_3$)

C → A ($C_2 \rightarrow A_2, A_3$)

C → B ($C_1 \rightarrow B_1, B_2$; $C_2 \rightarrow B_2, B_3$)

C → D ($C_1 \rightarrow D_1, D_2$; $C_2 \rightarrow D_2, D_3, D_4$)

B → A ($B_1 \rightarrow A_1$; $B_2 \rightarrow A_2$) (F.D)

B → C ($B_2 \rightarrow C_1, C_2$)

B → D ($B_2 \rightarrow D_2, D_3$)

Types of functional dependency

1. Fully Functional Dependency and Partial functional Dependency
2. Transitive F·D and No Transitive F·D
3. Single valued F·D and Multivalued F·D

name	address	course	fee	grade
p	Andheri	ADBMS	700	A
q	Parel	DBMS	800	B
r	Dadar	JAP	500	A
s	Thane	DBMS	800	A
sp	Andheri	TCS	900	A
t	Virar	TCS	900	B

Primekey attribute → attribute, part of primary key

Fully functional dependency → if we have any non primekey attribute which is used to functionally determine dependent upon all the primekey attributes, then it becomes fully functional dependency

Partial functional dependency → if any non primekey attribute can be identified using any of the primekey attribute, it becomes partial functional dependency.

Transitive F·D → if $x \rightarrow y$ and $y \rightarrow z$ than $x \rightarrow z$ is true for transitive F·D

and

Not Transitive F·D → if F·D is not transitive, it becomes No transitive F·D

id	sem	hostel
561	I	H-5
408	IV	H-4
819	I	H-5

eg id → sem
sem → hostel
than
id → hostel
(Transitive F.D)

Single valued F.D → for every value of X, we have exactly one value of Y.

Multivalued F.D → for every value of X, we have multiple values of Y.

Representation: X →→ Y
multivalued dependency

X →→ Y
X →→ Z

interesting
Here there should be atleast 3 to be attributes in a table

id	teacher	class	days
11	sam	comp	8
12	johan	comp	6
11	sam	elec	8
12	john	mech	6
13	mick	mech	2

id → teacher (single valued)
id → class
id →→ not there
(hence not multivalued)

interesting
class →→ teacher
class →→ days (multivalued)
both should match to atleast two attributes

Normalization

- 1NF
- 2NF
- 3NF
- BCNF
- 4NF

① First Normal form (1NF)
All the values present in a table must be atomic (single)

② Second Normal form (2NF)
(a) Table must be present in 1NF
(b) No partial dependency is present.

③ Third Normal form (3NF)
(a) Table must be present in 2NF
(b) Transitive dependency must be removed.

④ BCNF (Boyce - Codd NF)
(a) Table must be in 3NF
(b) All the determinants must be primary key.

⑤ Fourth Normal form (4NF)
(a) Table must be in BCNF
(b) Multivalued dependency must be removed

name	address	course	grade	fee	sem	hostel	duration
xyz	dadar	c_1, c_2	a_1, b_1	7000	VI	H7	10
xyz	dadar	c_3	b_1	8000	III	H3	11
pqr	parel	c_2	b_1	8000	V	H4	06
abc	andheri	c_2	b_1	8000	V	H5	09
abc	andheri	c_1	b_1	9000	VI	H3	09

value added for INF eg: all duplicate values are put in INF

INF → we decided candidate key as {name, course}

2NF → It must be in INF (true)
- No partial f.d dependency

Rule:
- $A^* \rightarrow C$ (f.f.d) → (A^*, B^*, C) (primary key)
- $B^* \rightarrow D$ (p.f.d) → (A^*, D^*)
- $E \subset B^*, E)$

where A^*, B^* are primary key respectively.

name → address (p.f.d)

course → fee, sem (p.f.d)

{name, course} → grade, hostel, duration (f.f.d)

①

name	address		
xyz	kalar		
PQR	pune		
abc	mathura		

ck = {name}

②

course	fee	sem
c_1	9000	VI
c_3	8000	III
c_2	5000	V
c_2	5000	V
c_1	9000	VI

ck = {course}

③

name	course	grade	hostel	duration
xyz	c_1	g_7	H7	10
xyz	c_3	b_1	H3	11
PQR	c_2	b_1	H1	06
ABC	c_2	b_2	H5	09
ABC	c_1	b_1	H9	09

ck = {name, course}

3NF → Table must be in 2NF

No transitive dependency should be present

Ex: $x \to y \to z$ → separate table for (x, y) and (y, z)

(1st and 3rd already in 3NF)

pk(course) → fee

pk(fee) → sem

④

course	fee
c_1	9000
c_3	8000
c_2	5000

⑤

fee	sem
9000	VI
8000	III
5000	V

After performing 3NF, we have ①, ③, ④ and ⑤.

BCNF → (1), (4) and (5) table are in BCNF.

(3rd) table is not in BCNF as

hostel → grade

is a functional dependency

AND

hostel is not a candidate key.

∴ (3rd) table is divided into 2 parts.

(6)

name	course	duration	hostel
XYZ	C_1	10	H_7
XYZ	C_3	11	H_3
PQR	C_2	06	H_4
ABC	C_2	09	H_5
ABC	C_1	09	H_3

(7)

(pk) hostel	grade
H_7	g_7
H_3	b_1
H_4	b_1
H_5	b_1
H_3	b_2

check!

duration → name

Assumed not in BCNF to show 4NF

pk = {name, course}

BCNF = {1, 4, 5, 6, 7}

4NF → multivalued dependency must be removed;

as need hostel key to relate with (7) else it will be lossy normalization

name →→ course

name →→ duration

name →→ hostel

(8)

(pk) name	course
XYZ	C_1
XYZ	C_3
PQR	C_2
ABC	C_2
ABC	C_1

(9)

(pk) name	duration
XYZ	10
XYZ	11
PQR	06
ABC	09

(10)

name	hostel
XYZ	H_7
XYZ	H_3
PQR	H_4
ABC	H_5
ABC	H_3

4NF = {1, 4, 5, 7, 8, 9, 10}

Transaction management and concurrency

The amount of work that is performed in a DB is measured by a unit called transaction.

Three operations are performed on transaction. These are:-

- Read(x) → transfer value of x from secondary to main memory and then reads the value of x.
- Modify(x) → changes or modifies the values of x present in MM.
- write(x) → the modified value is written back from MM to secondary storage.

T1

₹500 ₹1000 read(A); // 500

A ⟷ B A = A − 50; // 450

write(A); // 450

50

read(B); // 1000

B = B + 50; // 1050

write(B); // 1050

Transaction Properties (ACID)

- Durability
- Atomicity
- Consistency
- Isolation (Check if these are implemented using protocols)
- Interactive

Job of transaction manager

- Atomicity → all the operations must be performed completely or none of them are performed.

DB programmer

- Consistency → the state before and after the execution of transaction should be same.

(Summation (Σ(A+B)) must be preserved)

Concurrency Component

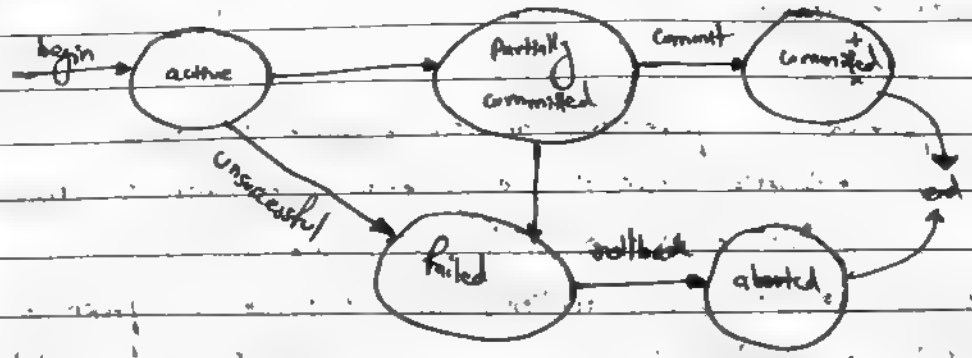
- Isolation → All the transactions are running in isolated manner without affecting each other.

Recovery manager

- Durability → Even if power failure occurs, DB is present in the system.

DATE / /

Transaction States



begin:- indicates start of transaction.

partially committed:- if transaction executed to the second last command of the transaction. (before commit)

failed:- transaction can fail in between or it can fail in partially committed state. Such transaction enter in failed state.

committed:- transaction successfully finished execution.

aborted:- failed transaction go in aborted state. Two operations are performed:-

1. Restart:- hardware/software failure
2. Kill:- data is missing

end:- finish the execution of transaction.

Page No.
Date / /

Schedules

- When two or more transactions are combined together, schedules are created.
- Schedules describe the order of execution of transactions
- Two types:-
  - i. Serial
  - ii. Concurrent → Always consistent
- Serial schedule → All the transactions are executed in serial manner. (one after another)

51

T1	T2
read(A);	
A := A-50;	
write(A);	
read(B);	
B := B+50;	
write(B);	
	read(A);
	A := A-100;
	write(A);
	read(B);
	B := B+100;
	write(B);

A=500, B=1000 → A=450, B=1050 → A=350, B=1150

Both are isolated

If serial schedule is there in the DB, consistency is always provided.

Two conditions:-
- eg:- → ① dependency
- ② without dependency

$\sum(A+B) = 1800$ → is preserved

also called Non serial schedule

Concurrency may or may not be passed

Concurrent schedules:—

Transactions are running concurrently using context-switching.

S2

T_1	T_2
read(A); //500	
A := A − 50; //450	
write(A); //450	
	read(A); //450
	A := A − 100; //350
	write(A); //350
read(B); //1000	
B := B + 50; //1050	
write(B); //1050	
	read(B); //1050
	B := B + 100; //1150
	write(B); //1150

is this most evolution?

(1050 + 450)

(1150 + 350)

S3

T_1	T_2
read(A); //500	
A = A − 50 //450	
	read(A) //500
	A = A − 100 //400
	write(A) //400
	read(B) //1000
write(A) //400	
read(B) //1000	
B = B + 50 //1050	
write(B) //1050	
	B = B + 100 //1150
	write(B) //1150

(1050 + 400) ✗

(1150 + 450) ✗

Not conflict

Not serial consistent

Serializability → process of converting non serial schedule to serial schedule.

Types:

(i) Conflict

(ii) View

Conflict → obtained by swapping non conflicting operations.

Rules for conflicting operations:—

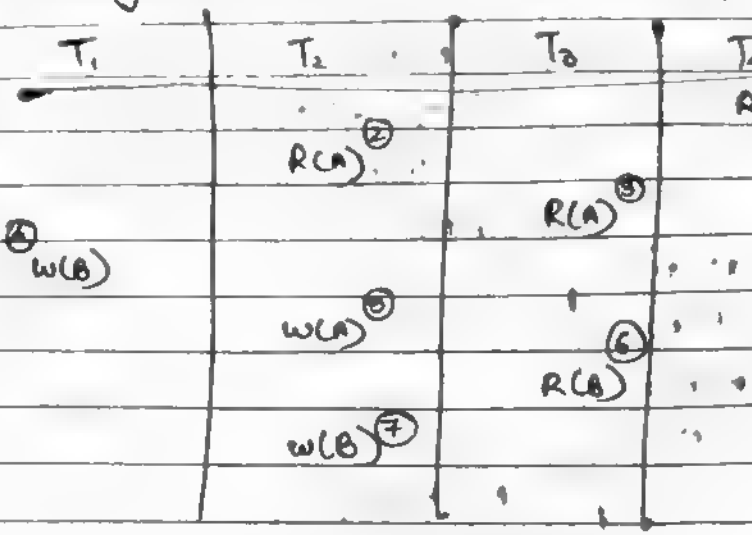
(1) both the operations must belong to different transactions.

(2) both the operations must be on same data item.

DATE / /

③ Atleast one of the two operations is write operation.

④ Check if the given schedule is conflict serializable or not



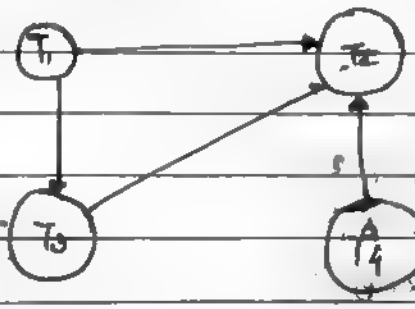
$T_1 \rightarrow T_3$, $T_1 \rightarrow T_2$, $T_1 \rightarrow T_4$

$T_3 \rightarrow T_2$, $T_3 \rightarrow T_1$, $T_3 \rightarrow T_4$

$T_2 \rightarrow T_3$, $T_2 \rightarrow T_1$

$T_1 \rightarrow T_3$, $T_1 \rightarrow T_2$, $T_1 \rightarrow T_4$

Precedence graph



As no cycle in the precedence graph, this concurrent schedule is conflict serializable.

S

T_1	T_2
R(A);	
A = A − 10;	
	R(A)
	Temp = 0.2 * A;
	W(A)
	R(B)
W(A)	
R(B)	
B = B + 10	
W(B)	
	B = B + temp
	W(B)

T_1	T_2
① R(A)	
	R(A) ②
	W(A) ③
	R(B) ④
⑤ W(A)	
⑥ R(B)	
⑦ W(B)	
	W(B) ⑧

As cycle, we do not have conflict serializability.

$T_1 \rightarrow T_2$

R(A) W(A)

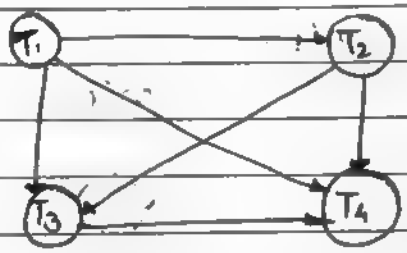
$T_2 \rightarrow T_1$

R(A) W(A)

R(B) W(B)

S

T_1	T_2	T_3	T_4
R(A)			
	R(A)		
		R(A)	
			R(A)
W(B)			
	W(B)		
		W(B)	
			W(B)



As no cycle, it is conflict serializable.

View serializability → Any non serial serial schedule is found to be view serializability if it satisfies following 3 rules:-

① If schedule is conflict serializable, then it is 100% view serializable.
Else it may or may not be view serializable.

② No blind-write → no 'view' serializability
If yes → may or may not be view serializability

③ Draw dependency graph → If cycle → no view serializability
Else if no cycle → view serializability

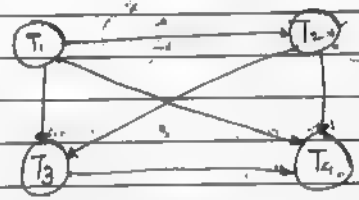
↳ we are checking two things (first write, final write)

Blind write → Transaction is not reading and directly writing blindly.

Q. Consider the given schedule S and check whether it is View serializable or not.

T_1	T_2	T_3	T_4
R(A)			
	R(A)		
		R(A)	R(A)
W(B)			
	W(B)		
		W(B)	
			W(B)

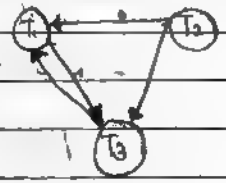
Step 1:-



As in precedence graph cycle is not present, it is conflict serializable therefore it is also view serializable.

Q.

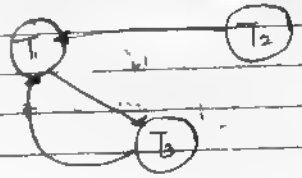
T_1	T_2	T_3
R(A)		
	R(A)	
		W(A)
W(A)		



As cycle is present, it may or may not be view serializable.

Step (2) As blind write is present in T_3, it may or may not be view serializable.

step ③



As cycle is present in dependency graph, it is not view serializable.

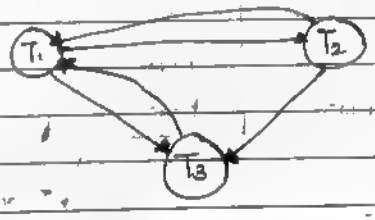
Q. T_1 T_2 T_3

R(A)

W(A)

R(A)

W(A)



step ① cycle is present, may or may not be view serializable.

step ② $T_2 \rightarrow$ blind write is present,

may or may not be view serializable

step ③

no cycle,

view serializable

Concurrency control in DBMS

In order to maintain consistency in DB, concurrency control is used. It provides locking concepts. There are two types:-

- shared lock (S)
- exclusive lock (X)

adv.

→ used for reading operation. Once shared lock is acquired, no exclusive lock (X) can be obtained. (But shared lock can be obtained).

→ used for reading and writing operation. If exclusive lock is obtained, no other lock can be acquired (No shared lock, No exclusive lock).

Can we obtain

If we have	S	X
S	True	False
X	False	False

T1 & T2

We have 2 accounts:-

A → ₹1000
B → ₹500

T_1	T_2
X-lock (A)	S-lock (A)
read (A)	read (A)
A = A - 500	unlock (A)
write (A)	S-lock (B)
unlock (A)	read (B)
X-lock (B)	unlock (B)
read (B)	display (A+B)
B = B + 500	
write (B)	
unlock (B)	

cascaded rollback

DATE / /

LOCKS ARE NOT SPECIFIC TO A PARTICULAR TRANSACTION

	T_1	T_2
1	X-lock(A)	
2	read(A)	
3	A = A − 500	
4	write(A)	
5	unlock(A)	
6		S-lock(A)
7		read(A)
8		S-lock(A)
9		unlock(A)
10		S-lock(B)
11		read(B)
12		unlock(B)
13		display(A+B)
14	X-lock(B)	
15	read(B)	
16	B = B − 500	
17	write(B)	
18	unlock(B)	
19	unlock(A)	

to prevent

disadvantages of unlock(A) at line 5:

- Consistency is not maintained
- Cascaded rollback — if unlock(A) is at 5 and there is an error in transaction of A, T_1 has to be rolled back. As S-lock(A) gets acquired as soon as unlock(A), it creates a cascading effect and T_2 has to be rolled back too.

By putting unlock(A) on line 19, we prevent both inconsistent state and cascaded rollback.

cascaded rollback — because of transaction T_1 failure, T_1 is rolled back but T_2 is reading the value which is given by transaction T_1 and therefore T_2 needs to be rolled back.

Case

T_1	T_2
X-lock (A)	
read (A)	
A = A - 500	
write (A)	
	S-lock (B)
	read (B)
	S-lock (A)
	read (A)
	unlock (A)
	unlock (B)
	display (A+B)
X-lock (B)	
read (B)	
B = 500	
write (B)	
unlock (B)	
unlock (A)	

In the above case, circular deadlock is seen.

S-lock (B) is read locked until, which prevents X-lock (B) from locking.

S-lock (A) cannot be locked as X-lock (A) is yet to be unlocked in T_1.

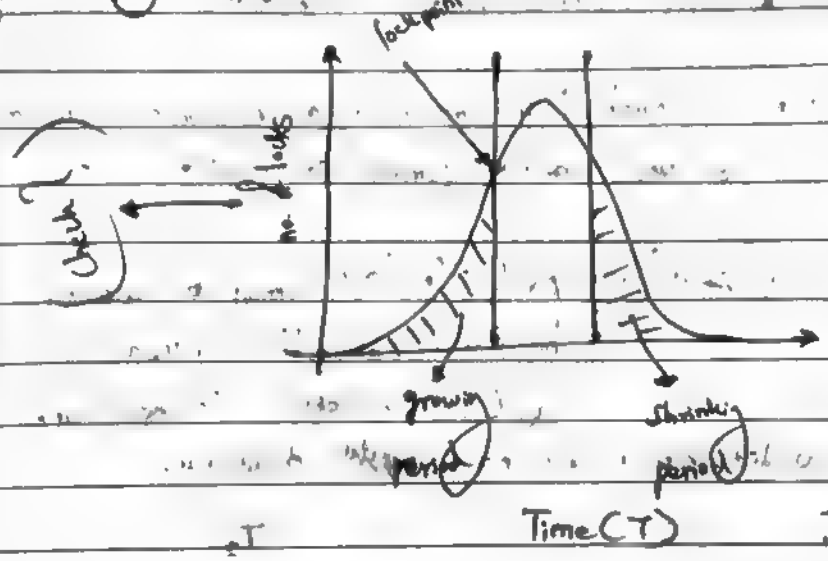
So we have deadlock.

2PL (2 phase locking) Protocol

Two phases are there:—
- growing phase
- shrinking phase

Growing phase → all the locks are acquired and none of them is released.

Shrinking phase → all the locks are released & none of them are acquired



lockpoint → point at which last lock is acquired.

Three types of 2PL:—

1. Strict 2PL → all exclusive locks are released only when transaction permits. (after commit)
   Shared lock can be released in between.

T_3	T_4
X-lock (A)	
read (A)	
X-lock (B)	
read (B)	

T_3	T_4
write (B)	x-lock (C)
S-lock (A)	
read (C)	
unlock (C)	x-lock (B)
Commit	read (B)
unlock (B)	write (B)
unlock (A)	

Strict 2 phase locking protocol does not suffer from cascaded rollback but it suffers from deadlock.

① Rigorous 2PL → Shared locks as well as exclusive locks are released after commit statement.

② Conservative 2PL (Static) → All the locks must be acquired at the beginning of the transaction and do not be released in between.

We have cascaded rollback but no deadlock.

T_1	T_2
x-lock (A);	
x-locks (B);	
read (A)	
A = A+50	
write (A)	
unlock (A)	x lock (A)
read (B)	read (A)
B += B+100	A = A+50
write (B)	write (A)
unlock (B)	unlock (A)

PAGE NO.
DATE / /

T T

Timestamp ordering protocol

In this the running each transaction timestamp value is given. Depending upon timestamp, the execution of the transaction takes place.

Two rules:—

① Transaction T_i issues read (Q) → not compared with read as read does not conflict with another read operation (serializable)

② If $TS(i)$ < write-timestamp (Q)
(transaction)

read operation is rejected and T_i rollback

③ If $TS(i) \geq$ write-timestamp (Q)

read operation is performed and read-timestamp(Q) = $TS(T_i)$

④ If T_i issues write (Q)

ⓐ If $TS(T_i)$ < read-timestamp(Q) OR

$TS(T_i)$ < write-timestamp (Q)

rejected, rollback

ⓑ otherwise accepted

write-timestamp(Q) = $TS(T_i)$

RTS(A) = 0 10.05
WTS(A) = 0 10.05

RTS(B) = 0 10.05
WTS(B) = 0 10.05

T_1	T_2	description
$T_s = 10$	$T_s = 10.05$	
read (B)	read (B)	$TS(T_1) \geq WTS(B)$
	b = b - 20	$10 \geq 0$
	write (B)	time(B) → 10
read (A)		$TS(T_2) \geq WTS(B)$
print (A+B)		$10.05 \geq 10$
	A = A + 30	time(B) → 10.05
	write (A)	$RTS(B) = TS(T_2)$
	print (A+B)	$= 10.05$

$TS(T_1) \geq WTS(B)$
$10 \geq 0$
$RTS(A) = 10$

$TS(T_1) \geq RTS(A)$
$10.05 \geq 10$
$RTS(A) = 10.05$

$WTS(A) = TS(T_2)$
$= 10.05$

As all the operations are performed concurrently under the given timestamp value, this schedule ensures freedom from deadlock.

RTS(A) = ~~0~~ 10.05
WTS(A) = ~~0~~ 10.05

RTS (B) = ~~0~~ 10.05
WTS (B) = ~~0~~ 10.05

T_1
$T_c = 10$

T_2
$T_c = 10.05$

TS (T_1) = 10 ← read(B)
RTS(B) = 0
WTS(A) = 0
RTS(B) = 10

read(A) → TS (T_2) = 10.05
WTS (B) = 0
RTS (A) = 10.05

b = b - 20;
write (B); → TS (T_2) = 10.05
WTS(A) = 0
RTS (A) = 10.05
Thomas Rule
WTS(B) = 10.05

read (A); → TS (T_2) = 10.05
WTS (A) = 0
RTS (A) = 0
RTS (A) = 10.05

a = a + 20;
write (A) → WTS(A) = 10.05
print (A+B);

read (A)
print (A+B);

TS (T_1) = 10
WTS(A) = 10.05
RTS(A) = 10.05

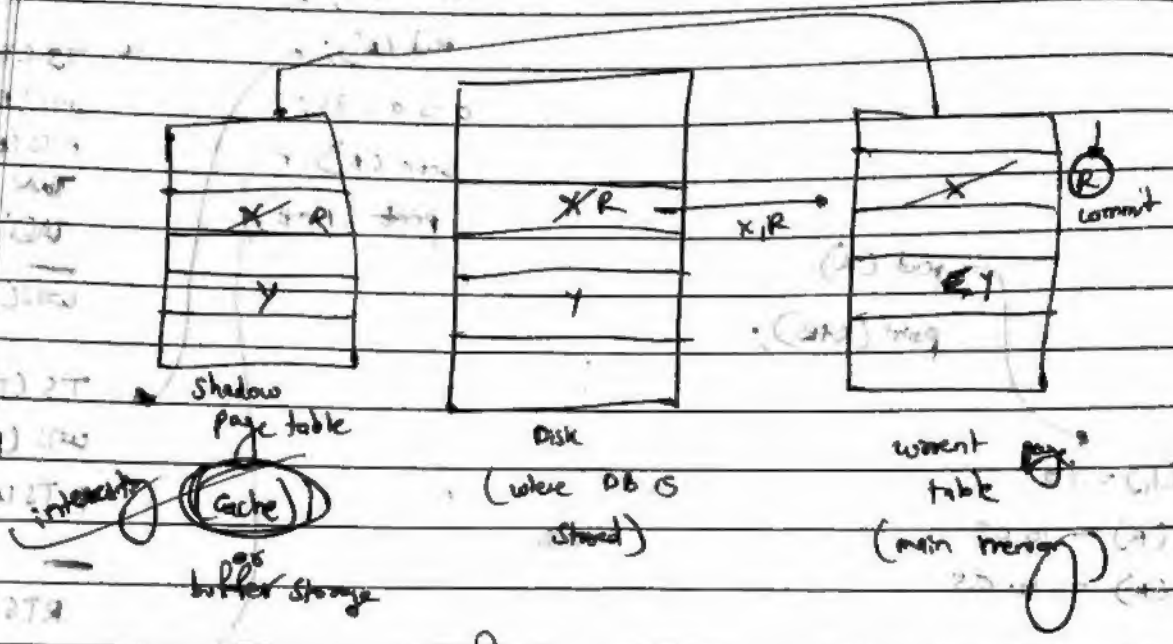
As WTS(A) > TS(T_1)
Cannot read
Rejected
Rollback

As T_1 is rollbacked,
T_1 is rejected and it has to start with new timestamp value.

Shadow paging

This is the alternative for log based recovery where two page tables are maintained:—

- shadow page table
- current page table



Here contents of current and shadow page table are same. If we want to perform operation on data item X, it gets transferred from disk to current page table, and current page table copies it to shadow page table.

If transaction is not committed, data can be recovered from shadow page table and if transaction is committed, Current page table copies data item to Shadow PT and Shadow PT transfers it to disk.

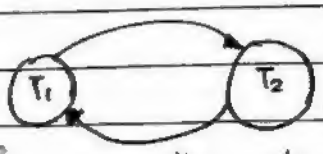
Deadlock

Deadlock is a situation where one transaction is waiting for another and vice versa. Therefore no transaction can proceed further with its normal execution.

Deadlock detection

deadlock detection is done using wait for graph (WFG) and we check whether WFG whether cycle is present or not.
If cycle is present, transaction is in deadlock state,
Else if no cycle, transaction is deadlock free.

T_1	T_2
X-lock(A)	
	X-lock(B)
X-lock(B)	
	X-lock(A)

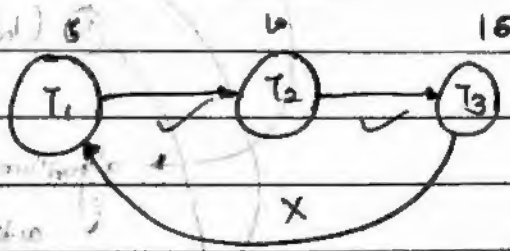


in deadlock state

Deadlock prevention

① wait-die → if $TS(T_i) < TS(T_j)$
→ T_i waits
else
T_i rollback

② wound-wait → if TS(T



$TS(T_1) < TS(T_2)$ (T_1 waits)
$TS(T_2) < TS(T_3)$ (T_2 waits)
$TS(T_3) \not< TS(T_1)$
(T_3 rolled back)
→ operation rejected with some random delay.

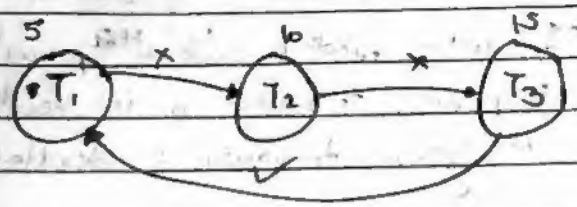
. With some random time as operation is accepted.

④ wound-wait → if $TS(T_i) > TS(T_j)$

T_i waits

else

T_j rollback



$TS(T_2) \neq TS(T_2)$

(T_2 rolled back)

$TS(T_2) \neq TS(T_3)$

(T_3 rolled back)

$TS(T_3) > TS(T_1)$

(T_3 waits)

ARIES → algorithm for recovery and isolation exploiting semantics

1. Analysis phase
2. Redo phase
3. Undo phase

→ algorithm checks for failure pages and set of transaction active in a DB.

→ it reapplies updates from the log to the DB

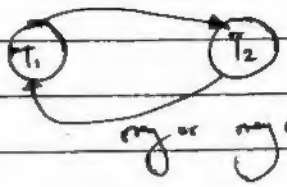
it is done only for committed transactions.

→ the log is scanned in backward direction and the no. of transactions which are active at time of system crash are restored with old values.

PAGE No.
DATE / /

Q.

T_1	T_2
R(A)	
	R(A)
W(A)	
	W(A)
R(B)	
	R(B)
W(B)	
	W(B)



may or may not be view serializable

no blind write,

not view serializable

Q.